Natural Language Direction Following for Robots in Unstructured Unknown Environments

Felix Duvallet

CMU-RI-TR-15-01

January 15, 2015

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Anthony Stentz, *chair*
J. Andrew Bagnell
Manuela M. Veloso
Nicholas Roy, *Massachusetts Institute of Technology*

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

To my parents,
who have always
supported me unconditionally
in all of life's many adventures.

Abstract

Robots are increasingly performing collaborative tasks with people in homes, workplaces, and outdoors, and with this increase in interaction comes a need for efficient communication between human and robot teammates. One way to achieve this communication is through natural language, which provides a flexible and intuitive way to issue commands to robots without requiring specialized interfaces or extensive user training. One task where natural language understanding could facilitate human-robot interaction is navigation through unknown environments, where a user directs a robot toward a goal by describing (in natural language) the actions necessary to reach the destination.

Most existing approaches to following natural language directions assume that the robot has access to a complete map of the environment ahead of time. This assumption severely limits the potential environments in which a robot could operate, since collecting a semantically labeled map of the environment is expensive and time consuming. Following directions in unknown environments is much more challenging, as the robot must now make decisions using only information about the parts of the environment it has observed so far. In other words, absent a full map the robot must incrementally build up its map (using sensor measurements), and rely on this partial map to follow the direction. Some approaches to following directions in unknown environments do exist, but they implicitly restrict the structure of the environment, and have so far only been applied in simulated or highly structured environments. To date, no solution exists to the problem of real robots following natural directions through unstructured and unknown environments.

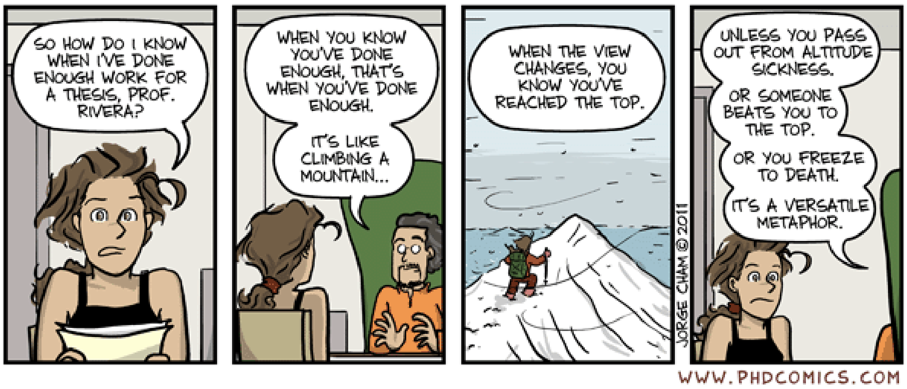
We address this gap by formulating the problem of following directions in unstructured unknown environments as one of sequential decision making under uncertainty. In this setting, a policy reasons about the robot's knowledge of the world so far, and predicts a sequence of actions that follow the direction to bring the robot towards the goal. This approach provides two key benefits that will enable robots to understand natural language directions. First, this new formulation enables us to harness user demonstrations of people following directions to learn a policy that reasons about the uncertainty present in the environment. Second, we can extend this by predicting the parts of the environment the robot has not yet detected using information implicit in the given instruction.

In this dissertation, we first show how robots can learn policies that reason about the uncertainty present in the environment. We describe an imitation learning approach to training policies that uses demonstrations of people giving and following directions. During direction following, the policy predicts a sequence of actions that explores the environment (discovering landmarks), backtracks when necessary (if the robot took a wrong turn), and explicitly declares when it reaches the destination. We show that this approach enables robots to correctly follow natural language directions in unknown environments, and generalizes to environments not encountered previously.

Building upon this work, we propose a novel view of language as a sensor, whereby we “fill in” the unknown parts of the environment beyond the range of the robot’s traditional sensors using information implicit in the instruction. We exploit this information to hypothesize maps that are consistent with the language and our knowledge of the world so far, represented as a distribution over possible maps. We then use this distribution to guide the robot, informing a belief space policy that infers a sequence of actions to follow the instruction. We find that this use of language as a sensor enables robots to follow navigation commands in unknown environments with performance comparable to that of operating in a fully-known environment.

We demonstrate our approach on three different mobile robots operating indoors and outdoors, as well as through extensive simulations. Together, learning policies and reasoning directly about the unknown parts of the environment provides a solution to the problem of following natural language directions in unstructured unknown environments. This work is one step towards allowing untrained users to control complex robots, which could one day enable seamless coordination in human-robot teams.

Acknowledgments


SO HOW DO I KNOW WHEN I'VE DONE ENOUGH WORK FOR A THESIS, PROF. RIVERA?
WHEN YOU KNOW YOU'VE DONE ENOUGH, THAT'S WHEN YOU'VE DONE ENOUGH.
IT'S LIKE CLIMBING A MOUNTAIN...
WHEN THE VIEW CHANGES, YOU KNOW YOU'VE REACHED THE TOP.
JORGE CHAM © 2011
UNLESS YOU PASS OUT FROM ALTITUDE SICKNESS.
OR SOMEONE BEATS YOU TO THE TOP.
OR YOU FREEZE TO DEATH.
IT'S A VERSATILE METAPHOR.
WWW.PHDCOMICS.COM

"Piled Higher and Deeper" by Jorge Cham www.phdcomics.com

In many ways, research is like climbing a mountain. Graduate school has been quite the journey, one that in many ways has paralleled my journey as a climber. I started climbing in earnest during my first year of graduate school, and by now the similarities between research and climbing are very apparent. Both have many ups and downs along the way, but at the end of the day the best you can do is work on one step at a time. The goal (be it a summit or new robot capability) is often clearly visible from far away, but the individual actions required to get there are far from obvious. Doing research and climbing a mountain require a lot of effort and planning, and usually involve plenty of suffering to get there. You can generally be assured things will not go according to plan, so remaining flexible in the face of bad results (or bad weather on a climb) has taught me a lot. That said, in my experience even the toughest adventures have always been worth it, and grad school has been no exception to this rule.

I believe it's the people you meet along the way that make all the difference. I have been lucky to work with some incredibly talented people during graduate school, beginning with my advisor Tony Stentz. Tony has been a great mentor on this journey, and his research vision and insights about challenging problems are unrivaled. Thank you for pointing the way, equipping me with the right resources and support, and letting me figure out the rest on my own.

The rest of my committee has been a great resource, from suggesting interesting research ideas to facilitating collaborations with people and robots. Thank you Drew, Manuela, and Nick for your incredible support throughout the years. Thanks also to Sanjiv Singh and George Kantor, you were important early mentors who introduced me to robotics research during my undergraduate years, and this shaped the path I am on today.

Partnerships are a critical aspect of both research and climbing, and I have been fortunate to work with (and learn from) some excellent roboticists through my graduate student career. This dissertation would not have been possible without excellent collaborations with (superscripts indicate chapter numbers) Alexander Grubb[3], Stefanie Tellex[3], Thomas Kollar[3], Matt Walter[5,7], Tom Howard[5,7], Sachi Hemachandra[5,7], Jean Oh[8], Abdeslam Boularias[8], and Bob Dean[8]. Looking back, I can safely say my most fruitful research was the result of these collaborations.

Seeing peers learn and struggle alongside me has been a great source of hope and comfort. Thanks to all of the Robotics Institute graduate students for the many hours working on assignments, projects, sitting through practice talks, or sometimes even relaxing. Thanks especially to Nathan Brooks, Michael Furlong, Scott Satkin, David Silver, Chris Skonieczny, Boris Sofman, Alex Styler, Breelyn Kane Styler, Nathan Wood, and the numerous talented people I have left off this list to save some room for the technical content: it has been great learning from (and with) you. The community of students at the Robotics Institute is incredibly strong, and this has been a big part of my experience. Thanks to Drew's LAIRLab group for accepting me as an unofficial member. Additionally, I would not be where I am today without countless hours spent in the CMU Robotics Club as an undergraduate, thanks especially to Steve Shamlian, Pras Velagapudi, and the many people I worked with as part of the Colony project.

Carnegie Mellon is a very special to do research, in large part because of the amazing resources available here. Thanks to the various organizations who have facilitated (or sometimes forced) social interaction: the Graduate Student Assembly, Dec5, and RoboOrg. The All University Orchestra and the CMU Explorers Club provided much-needed distractions. A special shout out to the Global Communication Center for running excellent workshops, and especially Doug Phillips for his many hours spent reading and suggesting improvements to this dissertation, and helping me distill complicated ideas into a clear and (hopefully) intelligible format.

I am deeply convinced the Robotics Institute would self-implode if it was not for the many amazing people who work tirelessly to reduce the entropy we graduate students bring, and I especially want to thank Cindy Glick, Sumitra Gopal, Alan Guisewite, Jean Harpley, Suzanne Lyons-Muth, Peggy Martin, and Sanae Minick. You keep the ship sailing so we can focus on fixing robots. NREC computing has also been a tremendous resource, especially when my laptop died less than a month before my defense (fortunately, all backed up). Truly, the resources afforded by CMU and the Robotics Institute have made this journey possible and even pleasant.

I hope to have contributed to this in a small way by starting the RI-meta seminar, a new series of talks dedicated to covering the higher-level (meta) topics we all use as researchers. Thanks to all of the inaugural speakers: Sanjiv Singh, Siddhartha Srinivasa, Jessica Austin, Illah Nourbakhsh, Matt Mason, and Manuela Veloso. I sincerely hope it has been as helpful to other students as it has been to me, and that these talks continue in the future.

Thanks to the many StackOverflow communities I have consulted over the years, especially the LaTeX Stack Exchange; designing – and typesetting – the figures and plots in this dissertation was a labor of love (emphasis on labor).

The last parallel between research and climbing is probably the most important: while the summit looks like the end of the journey, it is really the beginning. The best part about any adventure (be it research or a mountain) is not achieving that goal: it is what you learn on the journey and the partnerships you create along the way. In the end, it is all just a preparation for the next adventure. Onward!

A different kind of research: alpine climbing in Canada.

Funding

This work was supported in part by grants from several organizations, including the National Science Foundation under a Graduate Research Fellowship, the Office of Naval Research under MURI grant “Reasoning in Reduced Information Spaces” (no. N00014-09-1-1052), and the Robotics Consortium of the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program (Cooperative Agreement W911NF-10-2-0016).

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

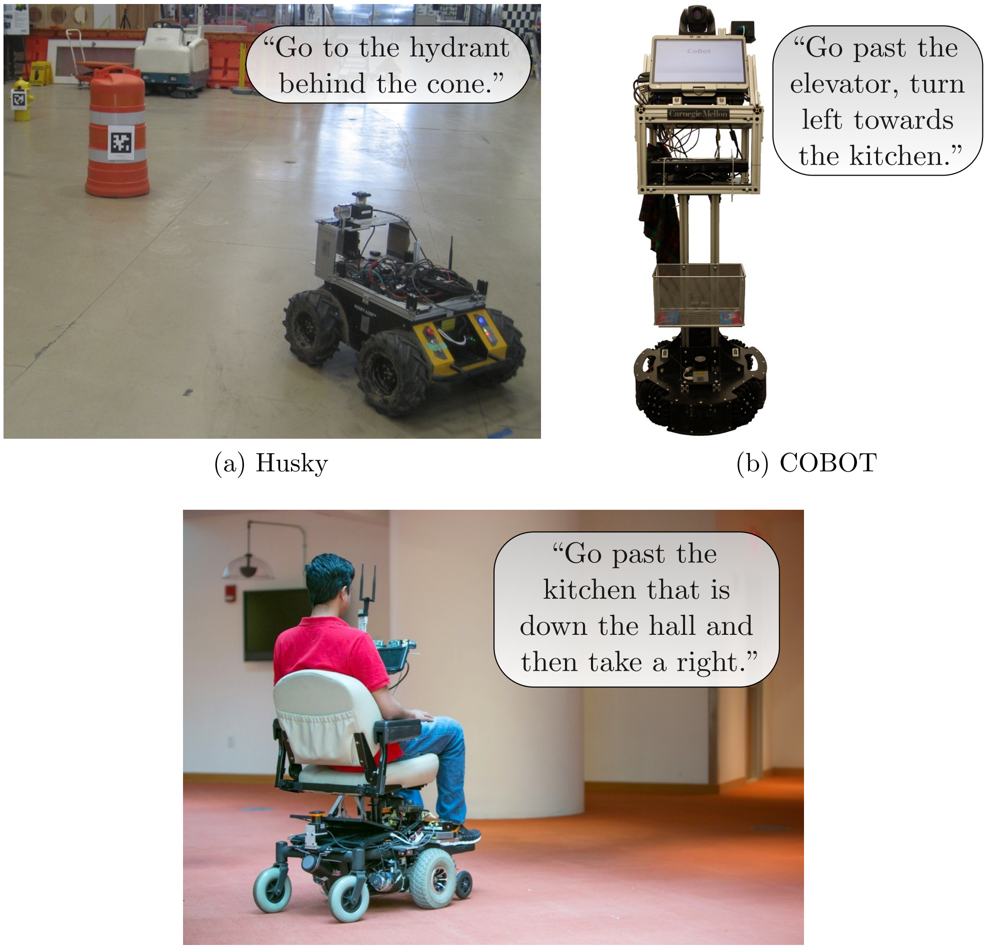
Introduction

> It is not the mountain we conquer,
> but ourselves.
>
> Sir Edmund Hillary

Robots are increasingly moving out of controlled isolation and into our homes and workplaces, where they will work alongside people on collaborative tasks. This increase in human-robot interaction brings about a much larger need for interaction modalities that do not require complicated interfaces, extensive training, programming knowledge, or specialized environments. Enabling robots to understand natural language instructions holds the promise of enabling lay users to control complex robots in an intuitive way, and could one day facilitate seamless coordination in human-robot teams.

One instance of a task where understanding language could bring about a flexible way to convey a complex behavior is navigation through previously unknown environments, where a person could direct the robot towards an unknown goal by describing how to reach it, just as they would explain to another person. For example, each of the robots in Figure 1.1 could be directed to a new destination using the language commands shown.

Prior approaches to this problem fall into two broad categories: approaches that require the map be fully-known ahead of time, and approaches that do not require a map but have only been applied in structured environments. Approaches that assume


"Go to the hydrant behind the cone."
"Go past the elevator, turn left towards the kitchen."
"Go past the kitchen that is down the hall and then take a right."

(a) Husky

(b) COBOT

(c) Autonomous wheelchair

Figure 1.1: Three different robots and sample natural language commands used in this thesis. Our goal is to enable robots to autonomously follow natural language commands given in unstructured unknown environments.

an annotated map is available a priori require collecting a complete map, which can be costly and time consuming. This is unrealistic in many scenarios, from robots operating in the home to disaster sites. On the other hand, approaches to direction following that do not require a map have so far only been applied to simulated or otherwise highly structured environments. This is because these approaches translate the command into a sequence of formal specifications of robot behavior and require the environment to match exactly what is expected by the specification; this makes these approaches ill-suited for complex maps and brittle to different environments. To date, no solution exists to the problem of following natural language directions through *unstructured* and *unknown* environments.

1.1 Thesis Problem

This thesis focuses on the problem of enabling robots to understand and follow natural language directions through unstructured unknown environments. We assume the robot has no prior map of the environment, nor do we make any assumptions about the complexity of the environment. In other words, a robot is put in an environment it has never seen or knows anything about, and is then given a natural language instruction that will bring it to a destination it has never been to. Following directions in *unknown* environments is an especially challenging problem because the robot only has access to a partial representation of the environment, one that is built up incrementally using the robot's perception system as it moves through the environment and receives sensor measurements. Traditional planning approaches of searching for complete paths that agree with the instruction are impossible without a complete map of the environment.

Instead, the robot must reason about a partial world model, and make a sequence of decisions that utilize the incomplete information available. One of the main challenges in this setting is that the actions or landmarks required for execution may not have realizations when the robot begins its execution. For instance, the robot's initial actions may be exploratory and appear to be incorrect. Furthermore, as the robot explores the environment it may take a wrong turn, and the robot must backtrack once it realizes it made a mistake. Finally, once the robot believes it has reached the destination it must explicitly declare it is done following the direction, even though there are still un-explored parts of the environment.

For example, consider the robot and instruction in Figure 1.1b. Given the robot's limited field of view, it may not initially be able to see the elevator so it must explore the environment to build up its partial map. As it is looking for the elevator the robot might appear as it were lost, and if it goes in the wrong direction and reaches a dead end without finding the elevator it must backtrack. When the robot eventually reaches the kitchen, it must explicitly decide it has finished following the command even though there might be another kitchen in the environment which an ambiguous command could have been referring to.

All of these challenges are in addition to dealing with complicated or vague or ambiguous spatial language, and operating in complex unstructured environments. While people are generally good at dealing with these trade-offs while following directions, autonomous robots currently do not have this capability. This thesis addresses these challenging issues, summarized in the following problem:

> **Thesis Problem:** Autonomous robots with realistic perception cannot yet follow natural language directions through unstructured unknown environments.

With this problem in mind, we now list the assumptions we make in order to restrict the complexity of the problem space.

Scope of Thesis Problem

We do not restrict the types of environments in which the robot can operate, indoor or outdoor, nor do we require that a map be available ahead of time. Our only requirement is that the robot is able to build a metric map of its environment as it operates, and can enumerate paths within it. We represent a combined semantic, topological, and metric map to efficiently reason about potential actions in the environment, but this is not a requirement of our approach.

Additionally, the robot needs to be equipped with a perception system that can perform object detection for a pre-defined set of object categories, returning both semantic labels and object geometry for the detected objects. As the focus of this thesis is not perception, our approach does not explicitly reason about detection likelihoods, confidence bounds, or false positives and negatives, although these can be present in the implementation. The robot need not be able to detect every possible landmark that could be described by the directions, nor do their labels need to match exactly what is mentioned in the directions (for example, a couch or chair could be detected as a "seat" by the robot). Obviously, any sensor available must respect physical constraints such as line of sight and maximum sensing range.

While we do not explicitly restrict the types of language commands that can be given to the robot, our work focuses on simpler commands that do not have complicated hierarchical clauses, negations, counting references, etc. We choose to focus on sequential task-constrained natural language directions that a person could reasonably follow in a new environment without a problem (for example, see the directions shown in Figure 1.1). Because of differences in perception capabilities (or viewpoints) between the person giving directions and the robot, the robot will still have to reason about ambiguity, as well as differences between landmarks names in the command and those returned by perception. Finally, our approach will assume access to people who are able to give and follow directions that are expressed in task-constrained natural language.

1.2 Summary of Thesis Approach

Because we are operating in unknown environments without access to a complete map, we frame the problem of understanding natural language route directions as inferring a sequence of actions in the world. This formulation as sequential decision making under uncertainty relies on a *policy* that predicts one action given the knowledge about the world so far. Executed on the robot, these actions can explore the environment (to discover new landmarks or regions of the environment), backtrack to a previously visited location (if the robot took a wrong turn), and explicitly declare when the policy believes it has finished following the direction. Each action moves the robot to a different location in the environment, which will build up its partial map of the environment.

While robots do not yet have the capability to follow natural language directions in unstructured unknown environments, people are quite good at it. We leverage this fact to learn the policy from people through *imitation learning*, using demonstrations of people giving and following directions. This formulation is especially well suited for unstructured environments, where engineering a system that can reason about the large number of variations present during direction following would be difficult and time consuming. Additionally, we are able to use a recent imitation learning framework to include examples that contain mistakes not present in the demonstrations. This enables the policy to learn how to recover from errors, for example, if the robot takes a wrong turn. Our results show that we are able to train policies that can follow complete directions through unstructured indoor environments, as well as anecdotal evidence of policies that generalize to different environments. We also apply our approach to operate in complex unknown outdoor environments, by learning in the space of planner cost functions.

We then extend this work to handle complex instructions that also convey information about the world (in addition to providing an instruction for the robot). For example, a direction such as "go to the kitchen that's down the hall" provides some information about the location of the kitchen, even though the robot may only be able to see part of the hallway. Utilizing this information is especially important in our map-less setting, as sensor range severely limits our knowledge of the world. We exploit the information *implicit* in the command to infer a distribution of possible maps that are consistent with the language and our knowledge of the world so far, effectively using language as a sensor that can build (uncertain) maps. This enables us to hypothesize the location of landmarks and "fill in" the unknown parts of the environment beyond the robot's sensor range. The policy then has more information to decide where to go, while still remaining flexible if our hypotheses turn out to be incorrect (for example, due to ambiguity in the language or perception errors). However, the policy must now reason in the belief space of the location of landmarks, and we present a novel belief space imitation learning algorithm that learns to reason about distributions using imitation learning and kernel distribution embeddings.

Research Agenda

We have addressed the following questions in our research agenda:

- How can autonomous robots successfully follow directions through completely unknown environments?
- How can we formulate direction following in partially-known environments as

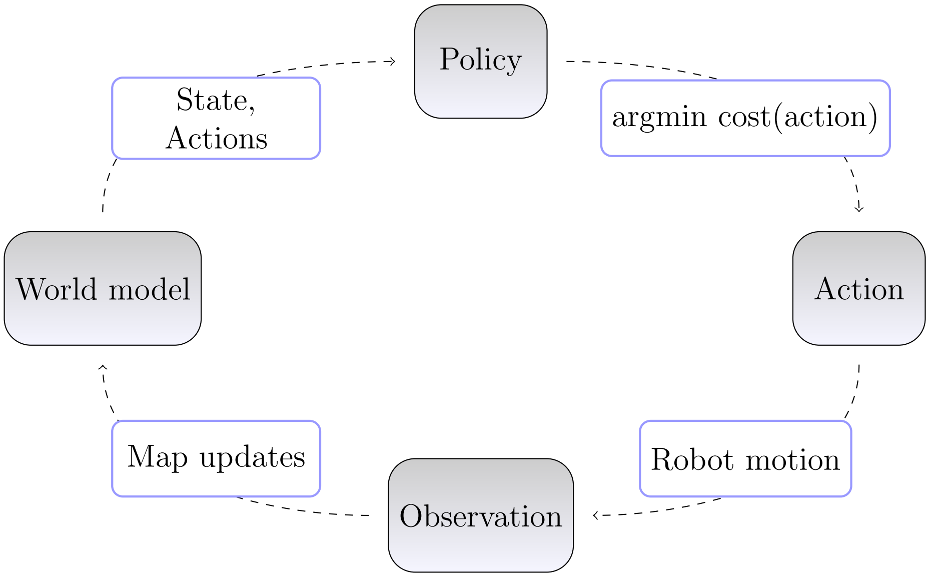


Figure 1.2: Basic approach formulation as sequential decision making: given a partial world model, we enumerate a set of actions that represent places we can go next. The policy evaluates all actions, and chooses the one with the lowest cost. The robot moves and receives a new observation about the world. This updates our partial world model, and we repeat the entire process.

sequential decision making so that a policy can efficiently make a sequence of decisions?

- How can we learn a policy from human demonstrations that can reason about the uncertainty present in unknown environments, recover from errors, and explicitly declare when it has finished following the directions?
- How can we exploit the information implicitly contained in directions to hypothesize maps that encapsulate the direction giver's knowledge about the parts of the environment that have not yet been observed by the robot?
- How can we extend our policy formulation to efficiently utilize this distribution of maps during decision making?

We now summarize the technical formulation of our approach.

1.2.1 Technical Formulation

Our formulation of direction following as sequential decision making relies on a policy π that will select a sequence of actions that best agree with the direction, given the policy's knowledge of the world so far. By introducing a cost function that measures the agreement with the direction, we can formulate this as minimizing the cost of the next action (out of all available actions) at time t, given the natural language instruction and the world known so far:

$$\pi = \underset{\text{actions}_\text{t}}{\operatorname{argmin}} \ \text{cost}\left(\text{action} \,|\, \text{language}, \text{world}_t\right). \tag{1.1}$$

Each action can be thought of as a "next place" to go, and actions can explore new parts of the environment, or backtrack to somewhere the robot has already been. Additionally, a special action declares that the policy has completed following the direction. Under this general formulation, the policy will cycle between selecting the lowest-cost action, moving in the world, receiving new observations, and updating its world representation, until the stop action is called. This formulation, shown in Figure 1.2, enables us to reason about a sequence of decisions instead of searching for a complete path, which would be impossible given the lack of a complete map of the environment.

To effectively follow directions, we now need four things:

1. A compact representation for partially-known environments (**state**).
2. An enumerated set of next places to move in the environment (**actions**).
3. A cost function that represents how well any state-action pair agrees with the direction (**policy**).
4. A way to learn this policy using demonstrations of people giving and following directions (**imitation learning**).

We represent the partially-known map at time t using a *semantic map* S_t that contains the location of all previously visited locations, as well as the semantically annotated objects that were previously detected. This map represents our knowledge of the world so far, and also keeps track of frontiers, which are locations that lie between the explored and unexplored space and could lead to new areas. The complete *state* of the system consists of this map, our current location x, and the linguistic utterance Λ.

We represent the set of *actions* A_t as paths in the environment: paths that end at frontier nodes are exploratory, paths that end at previously visited locations are backtracking actions, and a path that ends where the robot is currently is the stop action. Given the set of allowable actions, the policy evaluates each one according to a cost function c, and selects the action with the lowest cost:

$$\pi\left(x\right) = \operatorname*{argmin}_{a \in A_t} c\left(x, a \,|\, \Lambda, S_t\right). \tag{1.2}$$

To train the policy, we collect examples of people giving and following directions. We then learn the cost function such that it attempts to agree with the expert demonstrations over all training examples. This will be done using a loss function ℓ that penalizes disagreements between our policy π and the expert's policy π^* from the same state:

$$\ell\left(x, \pi^*, \pi\right) \propto \sum_{\text{examples}} \pi\left(x\right) \neq \pi^*\left(x\right). \tag{1.3}$$

Intuitively, if our policy agrees with the expert's policy in all possible states then we will not make any errors in direction following. However, since we are only given access to traces of the expert's policy (i.e. paths in the environment), we will need to derive the expert's policy π^* from these demonstrations. This will allow us to include training examples of recovering from failures (even if the demonstration didn't include such an example), and enable the policy to learn how to recover from mistakes.

Rather than reason over the entire instruction at one time, we exploit the sequential nature of spatial language to decompose the problem into a sequence of several simpler problems:

$$\Lambda \to \left[\Lambda_0, \ldots, \Lambda_N\right], \tag{1.4}$$

where the policy now only has to reason over a single element Λ_i in the sequence. This allows us to require Equation (1.2) as:

$$\pi\left(x\right) = \operatorname*{argmin}_{a \in A_t} c\left(x, a \,|\, \Lambda_i, S_t\right). \tag{1.5}$$

Furthermore, we will extract a structured clause from each sub-utterance, enabling us to understand the meaning of the command. More specifically, this enables us to understand the verb that was commanded, any landmarks mentioned in the utterance, and spatial relationships between the desired path and the landmark in the direction.

A key insight of this work is that each utterance Λ_i contains two types of information: *explicit* information that tells the robot where to go, and *implicit* information that may not be strictly required for correctly following the direction yet conveys useful knowledge about the environment. We exploit this implicit information in Λ (in addition to the history of sensor observations z^t) to generate a distribution over semantic maps: $p\left(S_t|\Lambda, z^t\right)$. This is essentially representing a distribution over possible environments, where the area beyond the robot's sensor range is hypothesized. While the policy formulation remains essentially unchanged, it must now reason over a *distribution* of semantic maps:

$$\pi\left(x\right) = \operatorname*{argmin}_{a \in A_t} c\left(x, a \mid \Lambda_i, p\left(S_t|\Lambda, z^t\right)\right). \tag{1.6}$$

The policy embeds the distribution of semantic maps using a kernel distribution embedding. This solution remains efficient in the presence of map uncertainty, and we can apply our imitation learning formulation in this space.

1.2.2 Thesis Statement

Together, the formulation as sequential decision making under uncertainty combined with the ability to reason directly about the unknown parts of the environment provides a solution to the problem of direction following in unstructured unknown environments. Our thesis statement follows:

Thesis Statement:
Following natural language directions through unstructured unknown environments can be formulated as sequential decision making under uncertainty, and can be solved using a policy learned from human demonstrations.

Furthermore, language can be used as a sensor; this enables robots to infer a distribution of maps that extends beyond their perception range. By extending the policy to reason in belief space, robots can follow complex natural language directions in unknown environments with performance that approaches operating with a known map.

This thesis enables robots with realistic perception systems to autonomously follow directions expressed in natural language through unstructured unknown environments. This work is one step towards allowing untrained users to control complex robots, which could one day enable seamless coordination in human-robot teams.

1.2.3 Metrics

To evaluate the performance of our system, we will use several metrics. Most importantly, we will measure the ending distance from the correct final destination. We consider the resulting path for a direction *successful* if it ends within some distance of the correct destination (for example, within 5 m). This is the primary success metric, as following directions correctly is the main goal of this thesis. We will also measure the path length taken to reach the destination (compared to a direct path), which enables us to evaluate the effect of exploration and backtracking. Minimizing extra distance traveled is a secondary goal.

By ablating parts of our direction following system, we will compare our approach against different baselines to quantify the importance of the various components in our system. For example, we can consider complete and incomplete semantic maps, full and partial feature spaces, or following directions with and without belief space reasoning. These will provide insights into what is most important when designing a direction following system for unknown environments.

1.3 Thesis Outline

We begin this dissertation by reviewing prior approaches to similar problems, as well as relevant related work in a variety of disciplines in Chapter 2. We then dive into our technical approach: we first show in Chapter 3 how natural language direction following in unknown environments can be formulated as sequential decision making under uncertainty, where a policy predicts the next best action to take given the world seen so far. This formulation relies on a policy that is able to evaluate the cost of all available actions, and selects the one with the lowest cost. While this policy is compact and has a simple form, it must take into account all relevant aspects of the direction, environment known so far, and the action to take. This is encapsulated in our feature representation, which we will describe in more detail. We also describe the properties of spatial language that enable us to decompose this problem into smaller sub-problems. We then show how to learn the policy through imitation learning, using demonstrations of people both giving and following directions in Chapter 4. In that chapter we also provide results of our approach to following natural language directions in unknown environments.

Expanding upon this work, we then propose a novel view of language as a sensor that can be used to infer maps beyond the range of traditional sensors (such as cameras and laser range finders) in Chapter 5. While this approach enables us to use more of the information contained in the natural language direction, it generates a distribution of maps that the policy must now reason about. To address this, in Chapter 6 we describe extending our policy to reason over the space of map distributions, and also how to learn a belief space policy (again using imitation learning).

A major strength of this work is its broad applicability to mobile robots across a variety of platforms, sensor suites, and environments. Chapters 7 and 8 describe integrated demonstrations on the three robots shown in Figure 1.1, operating both indoors and outdoors. We finish in Chapter 9 with a summary of this dissertation, highlight its contributions to the field, and discuss avenues for future research.

Chapter 2

Background

> One day's exposure to mountains is better than a cartload of books.
>
> John Muir

This work draws upon – and is inspired by – state of the art research in many different areas, both within and outside of robotics. We first relate our problem to prior approaches to natural language understanding for robots (Section 2.1), and highlight the novelty of our problem. As the focus of this thesis is primarily on direction following, we will focus on approaches to this problem. We then discuss related work in a variety of other areas that will be applicable to our solution (Sections 2.2 to 2.5): we will draw from work in imitation learning, belief space planning, active exploration of unknown environments, and semantic mapping.

2.1 Natural Language Understanding for Robots

While people naturally communicate and collaborate using language, for the most part complex robots are still operated by users with extensive training, through programming instructions, or with very specialized interfaces. If robots are ever to be ubiquitous and controllable by lay users, we must introduce new modalities for controlling complex autonomous systems [162].

In collaborative tasks, people have been shown to prefer communicating using speech, whether it is compared against written and typed instructions [110] or other traditional computer-based modalities [134]. While speech as an input modality has been studied for computer systems such as speech-to-text transcription, personal digital assistants (e.g., Siri and Google Now), automated phone dialogue, and many others, the majority of applications to date have not been physically situated: speech and natural language have not yet become widely used to control robots. A significant barrier for robots to understand natural language lies in the symbol grounding problem: connecting symbolic objects with their corresponding real world counterparts in the environment [54]. For grounding in natural language directions through an environment, the symbols (linguistic terms) for objects, verbs, and spatial relationships must be mapped onto the corresponding real world objects and actions.

In this section we will focus primarily on work towards understanding natural language directions. Specifically, we will review work on understanding *spatial language* directions, which describe (in natural language) the actions necessary to reach a destination. Within this field of work, approaches have tackled two main sub-problems: (1) direction following in unstructured (but known a priori) environments, and (2) direction following through unknown (but structured) environments. Approaches to these two main sub-problems will help us better understand the main challenges of following directions through unstructured and unknown environments, and will inform much of our approach in this novel problem space.

2.1.1 Following Directions through Known Environments

When a complete map of the environment is available a priori, it is possible to search for complete paths (from the robot's starting location to any possible destination) that match the natural language instruction. Since the entire map is available, it is possible to optimize the entire trajectory globally, utilizing all possible landmarks in the environment and the entire direction.

The actual details of approaches to this problem vary from sequencing hand-tuned action primitives, inferring formal controllers or reward functions, applying reinforcement learning, or structuring the problem as inference in a graphical model. Because they all assume a known map and perform global inference, they are able to reason successfully about natural language directions in very unstructured environments.

Engineered Action Primitives

It is possible to use action primitives that describe the intent of the natural language direction, represented as a set of rules that can be executed by the robot (or planner). One instance of this is work by Levit and Roy [86] that maps language to basic semantic components called *Navigational Information Units* that are essentially hand-generated rules describing the desired shape of the path. These procedures include moving around an object, moving to absolute locations in the map, turning, and verifying closeness to a given landmark. Complete paths are generated from these semantic components using a Dynamic Programming approach. The authors created a system that can follow directions in the MAP-TASK scenario, a corpus of maps and natural language directions [5]. A very similar approach has also been used to generate paths in known indoor environments [44]. Our approach bypasses the need for this intermediate representation, reducing the amount of engineering effort required.

Route Graphs are a similar formalized representation of the natural language command that use a sequence of parameterized rules that represent desired turns, landmarks where these turns can take place, and goals [106]. In fully-known environments, these sequences can be combined using fuzzy rules and evaluated for entire paths using search trees [95] or particle filters [84]. For the most part, evaluating the fitness of a path requires hand-tuned functions for each possible rule. Recent work by Landsiedel et al. [84] focuses on automatically learning to label places in the environment and a probabilistic model for the route descriptions.

Formal Controller

Another representation for the desired robot behavior are plans based in formal logic, studied by Kress-Gazit and colleagues [76, 77]. Their approach translates structured natural language into formal logical specifications of the task (in this case Linear Temporal Logic), which they use to generate controllers that will execute the desired command. This approach of using formal logical specification provides verification guarantees on task success if the task is feasible, and can generate explanations when a task is unachievable [87, 120]. However, doing so requires a complete map of the environment and the types of input commands in these approaches are closer to a programming language than natural language.

Reward Functions

Instead of mapping language to individual components such as action primitives or controllers, other approaches reason at a more global level by converting language to reward functions that are optimized using a planner to generate the desired behavior. Recent work has investigated translating instructions in English to a reward function in a Markov Decision Process (MDP) with formal action specifications [89, 90]. This approach uses statistical machine translation and Expectation Maximization and the authors have applied this to simple tasks in abstract known environments. Users are able to specify high-level natural language instructions, then a simulated robot translates the command to abstract reward functions. Our approach also learns a reward function (as part of the policy), but their approach requires a complete map and MDP formulation of the problem. Furthermore, the action specifications used in this approach are essentially the same as the action primitives above, so this approach still requires engineering effort to design the actions and their specific reward functions. Our approach is more general in that it does not require a world map or action primitives.

Policy-Based Approaches

Along the lines of learning reward functions for following directions, some approaches learn the value function in a *policy*, using reinforcement learning. For example, Vogel and Jurafsky [167] used Reinforcement Learning in the MAP-TASK corpus to learn a policy that grounds both spatial and linguistic components of the directions. They define a state and action representation that encompasses landmarks and cardinal directions, and learn the correspondence between the language and features of the path through a value function. This value function can then be used by a policy to follow directions. They assume complete knowledge of the map during training and testing. Our work is similar to this approach in a few respects: we use a policy to choose a sequence of actions, and compute features of paths to generalize to new scenarios. However, the MAP-TASK scenario only provided a discrete number of well-identified landmarks [5] so both of these approaches place constraints on the number of possible actions, making this a structured known environment.

Andreas and Klein [8] present an extension of this work to learn a grounded representation of the path directly from the language. Although this work requires a complete map of the environment, it does not limit the possible action space or rely on pre-specified procedures. They are also able to learn groundings for other interesting tasks, such as naming colors and textually describing the time series behavior of stock prices. Like these reinforcement learning based approaches, we will learn the meaning of words using a feature representation, but we will do so in an unknown environment with many more possible actions.

Graphical Model Formulations

Some approaches attack the grounding problem almost literally by formulating the problem of following directions as inference in a graphical model. For example, by treating direction following as sequence labeling, Shimizu and Haas [138] train a Conditional Random Field (CRF) that maps instructions to an action sequence through a known environment. Although we also treat direction following as a sequential prediction problem, our policy does not have access to a complete map. Furthermore, their approach only utilizes a graph representation of the world (with no landmarks), so the types of directions that can be represented is restricted.

The state of the art end-to-end direction following system is the recent work of Kollar, Tellex, and colleagues that also formulates the problem as inference in a graphical model [61, 73, 74]. Their approach extracts semantic structures called Spatial Description Clauses (SDCs) from language and plans a complete path through a known environment by grounding actions, places, and landmarks in the language to their respective components in the semantic world representation. A graphical model called the Generalized Grounding Graph (G^3) then infers the complete path that best corresponds to the instruction. This approach uses a set of spatial features to reason about the relationships between paths and landmarks [157], and several linguistic components to reason about the similarity between objects in the world and landmarks described in the command [72] that uses WordNet [45, 101]. Recent work extends the grounding graph to include paths and events, which can be used for mobile manipulation [158].

This approach is related to ours in that we share the same semantic descriptions of language (SDCs) and some features, but we frame direction following using a fundamentally different approach. While their approach is to plan a *path* through a known environment, our approach is explicitly designed to operate in unknown environments by learning a *policy* that reasons about uncertainty and generates a sequence of short-horizon actions (reasoning about the information currently available). Additionally, by training directly in unknown environments we learn a policy that can recover from mistakes by backtracking.

While their approach assumes a complete labeled map is known a priori, Kollar et al. [73] present very preliminary results using their globally-trained model on partial map information that is built up as the robot moves in the environment, using a greedy local inference algorithm. This approach (with no prior map) yields significantly worse performance on their corpus of directions than the global search [73]. This is likely due to a fundamental mismatch between the data available at training and validation time: since this approach is trained on entire paths it is unlikely to make the correct decision when no options appear to be correct. For instance, a direction like "go to the elevators" is unlikely to result in the correct partial path when the elevators have not been detected, since the algorithm has never been trained on what to do when landmarks are not visible. Our approach does reason about this possibility because it has been trained to do so, and chooses the action that *best* matches the direction using the information available. Furthermore, our results show that while access to a fully-labeled map does sometimes improve the performance of our approach, the difference in our work is not significant.

Another formulation of natural language understanding is introduced by Howard et al. [60] as inferring planning constraints. This approach uses a novel graphical model (the Distributed Correspondence Graph) to map from language to a set of constraints that obey the instruction. A trajectory optimizer uses these constraints to compute the final path. We will make use of this graphical model in later chapters of this work and our policy will be used as their trajectory optimizer.

Summary and Comparison

Following directions in completely known environments is different than its unknown environment counterpart, in part because the approach can reason over the entire path and is able to use global information about the world and the direction. For example, knowing the direction ends near an uncommon landmark in the environment constrains the search to paths that end near that location. Additionally, being able to do an exhaustive search in the environment ensures no alternative is missed (which can occur in partially-known environments). This provides a level of robustness, and on the whole these approaches are able to handle complex unstructured environments quite well. However, collecting a complete annotated semantic map is time consuming and may be impossible in some scenarios such as disaster response.

Keeping this important limitation in mind, we do gain several important insights from this class of approaches that will become useful in our problem of following directions through unstructured *unknown* environments:

- we will represent the world using a combined metric, topological, and semantic map,
- our policy will learn the meaning of action words directly without an intermediate representation, and
- we will utilize a feature-based representation of the world in order to generalize to new scenarios.

We will represent the world known so far using a partial map, and since we cannot optimize complete paths we will use this partial map to enumerate actions that represent potential paths towards the destination. Given these candidate actions, a policy will predict a sequence of decisions that move the robot in the world, following the direction. To do this, the policy will evaluate all actions (using features extracted from each path) and selects the one that best matches the direction under the map known so far. This process will be repeated until the policy decides it has reached the destination.

2.1.2 Following Directions through Unknown Environments

The approaches presented so far treat direction following as inferring a complete path through an environment and thus require access to a complete semantically annotated map of the environment. This assumption is not always realistic or practical, as collecting a full map may be expensive, time consuming, or simply impossible in many domains. Enabling robots to understand natural language in *unknown* environments would allow them to operate in many more situations.

We now present approaches that have investigated this problem of direction following through unknown environments. Because they cannot reason globally, these solutions make a sequence of decisions, using only partial map information that has been collected so far. To date, all approaches in this category share a common theme: they map language to a formal intermediate representation of the intent of the direction, then execute it on the robot open-loop. There are two ways of mapping from language to this intermediate representation: using engineered rules or a learned parser. As we will see, these approaches are able to successfully follow directions through previously unknown environments. However, because of the assumptions made in the formal intermediate representation, so far these solutions have only been applied in highly structured environments.

Engineered Approaches

The first class of approaches map language to a sequence of hand-generated parameterized action primitives such as `travel()` or `move_forward_until()`. Representing the natural language direction using these primitives does not require access to a complete map.

MARCO is a system that follows free-form natural language directions through an unknown virtual environment [91, 92, 93]. This approach uses a hand-coded parser to sequence parameterized action primitives (called *procedural specifications*) from the natural language direction. These local procedures consist of several action primitives, such as moving, turning, verifying the presence of an object, and declaring the goal has been reached. These procedures also include pre- and post-conditions that specify what the instruction follower (the robot) expects to see before and after the travel procedure. This can be used to infer implicit actions that must be performed in addition to the explicit commands. This work was an impressive implementation of an end-to-end simulated system that could handle a wide range of natural language commands, but it operated entirely in a highly-structured game-like environment. This environment consisted of regular grid-aligned hallways and intersections, along with a limited set of highly-discriminative landmarks. These types of highly-structured environments are very different from the real world indoor and outdoor environments in which robots will be expected to understand natural language.

Using a very similar approach, the Instruction Based Learning project collected a corpus of directions for a small robot operating in a miniature model of an urban environment [24, 81, 85]. Using pre-programmed primitives, the authors mapped language onto sensory-motor behaviors, such as moving towards an intersection or turning at a given location. Because their action primitive set is closed, the authors note this can lead to robustness problems. This approach also does not reason about uncertainty and cannot backtrack if it makes a mistake. This body of work provided a corpus of directions and developed basic action primitives to sequence together. However, the environment was a simple city-like simulation and the robot primarily used road intersections to navigate. Our approach operates in a variety of indoor and outdoor environments and can use arbitrary landmarks.

While these systems were successful in following directions through unknown environments, they operates solely in structured environments with a small set of possible actions or landmarks. Furthermore, they rely extensively on a collection of hand-tuned rules and engineered components instead of learning the meaning of commands directly. Our approach aims to operate in more unstructured environments that do not have clear regular structure, which precludes the use of hand-programmed robot behaviors.

Learned Approaches

Instead of using hand-generated rules for parsing language, several approaches instead learn the mapping from natural language to robot controllers. A key benefit of learning a parser is the ability to handle more complex commands with varied input language.

For example, approaches presented by Matuszek et al. [99] train a language parser that can be used to follow natural language directions through both known [98] and unknown [99] environments. This parser can map language into formal procedural plans expressed in a LISP-like formal specification called Robot Control Language (RCL). These procedures are then executed by the robot in the environment. However, these particular approaches only take into account the topology of a building (hallways, rooms, and intersections), without using any landmarks for navigation [99]. Our approach reasons both about the structure of the environment (topology) and landmarks within it (semantics).

Similarly, Chen and Mooney [29] extended upon the work of MacMahon [92] to learn a semantic parser and a lexicon for representing directions in a formal navigation plan language, using training data consisting of example directions and paths. Even though this approach can handle a wider range of commands, it is still restricted to use the same local procedures. In this approach, following the directions will fail if the environment does not match what the action primitive expects. This approach was validated on the same simulated (highly structured game-like) world developed by MacMahon [92], imposing the same requirements on the structure of the environment.

Summary and Comparison

These presented approaches to the problem of following directions do not assume access to a complete map of the environment. They engineer or learn a mapping from the natural language instruction to a sequence of action primitives, and are then executed by the agent in the environment open-loop. The intermediate representations range from parameterized action primitives to formal LISP-like statements, and they represent a formal description of the instruction in the form of a robot-executable controller.

Having an intermediate representation of the intent does enable this class of approach to handle some complex language commands (e.g., “take the second left,” “go until you reach an intersection”). However, mapping language to robot controllers leads to a key assumption for this class of solutions that the robot will be able to execute the desired controller perfectly and the environment will match what the controller expects. The formal symbolic representation of the command in these approaches is rigid and can lead to brittleness, for example when there is a large mismatch between the environment and controller expectations.

As a result, these existing approaches all make simplifying assumptions about the structure of the environment, from assuming highly-regular simulated environments to modeling indoor environments as a topological graph without any landmarks. Furthermore, all of these approaches have been validated in simulation only, or by making strong assumptions on the structure of the environment. It is unlikely that a real robot exploring the environment would generate a partial semantic map that is structured like the environments these approaches currently operate in. Indeed, maps generated by robots during execution in unstructured environments are generally noisy: hallways are not exactly straight, intersections are not always right angles, etc. This restricts the complexity of the environments that can be handled by this class of solutions.

The environment limitations required by the above prior approaches mean that they have only been applied to *structured* unknown environments. Keeping these limitations in mind, we do gain two important insights that will be useful in our problem of direction following through *unstructured* unknown environments:

- our policy must make a sequence of decision using local information only, and
- our policy must explicitly reason about when it has finished following the direction (stopping).

We remove the reliance on action primitives or intermediate formal representation of the language; in our approach we instead train a policy that reasons directly about the meaning of words. This representation is more flexible, and as we will show can handle unstructured unknown environments.

2.1.3 Other Related Language Understanding Problems

We now review some of the relevant literature on enabling robots to understand natural language in settings *other* than direction following. For example, some of the interesting application areas are dialogue systems, understanding general (non-navigation) instructions, and language-driven human-robot interaction or semantic mapping.

Dialogue and Language Generation

Dialogue between people and physically-situated systems (such as robots) poses special challenges, such as engagement (when to start or stop a conversation) and turn-taking (when to speak during a conversation) [19, 25]. Several approaches attempt to solve the inverse of our problem: that of *generating* a natural language description for a path, task, situation, or help request. Solutions to this problem include a game-theoretic approach that model the rationality of the speaker and listener [53], formal logic specifications of plans [120], inversions of an existing probabilistic graphical model [160], information-theoretic human-robot dialogue modeling to generate queries that would reduce the entropy in the groundings [38, 159], inverse reinforcement learning from human demonstrations [113], and Monte-Carlo simulations of a direction-follower to maximize the likelihood they reach the destination [52]. These approaches all map from a desired action (for example a specific action to be performed by an operator) to a natural language description of the task, with the idea that a person could then execute the desired task.

Understanding General Instructions

One of the earliest approaches for understanding natural language instructions was presented by Winograd [170], where a dialogue-driven interface could be used to manipulate objects in a simulator.

Agre and Chapman [3] introduce the notion of “plans-as-communication,” where a set of natural language directions serve as “guides to activity” and exploit shared understandings of the world. The directions provide a skeleton of a plan that requires improvisation, instead of a complete solution to the problem [3, 27].

More recent work has focused on understanding general instructions for everyday manipulation tasks by mapping high-level instructions in natural language to a sequence of robot primitives similar to the ones present in the direction following literature. This has been applied to tasks ranging from setting a table [161], completing mobile manipulation tasks in the kitchen [102], or following a Microsoft Help instructions [21, 22].

Other related work has investigated *teaching* tasks using natural language. Work in this area generally uses language to sequence together smaller subtask components the robot may already know. Approaches to this problem include interactively defining new tasks using keywords and basic primitives [100], learning formal pre- and post-conditions for new tasks described in natural language [26, 43], combining dialogue with observations of human behavior to teach new tasks to a robot [137], or using dialogue to add, modify, or cancel tasks by interacting directly with an autonomous robot [154].

The ability to teach arbitrary tasks to robots using a combination of natural language, gestures, demonstrations, and other sources of prior knowledge (for example the web) will one day enable robots to become effective teammates in human-robot teams.

Language-Driven Human-Robot Interaction

Language has been used as one component of a multi-modal interface for controlling robots. Speech is an especially motivating interaction modality as it requires essentially no user training, and could be used for very short-term interaction between robots and people. These systems range from robots that can interact with passersby in crowded environments in order to reach an unknown destination, to single-user interaction in controlled settings.

GRACE was an early integrated fielded system that successfully navigated to a registration desk in a previously unknown environment by asking people for help and following simple directions [145]. However, the natural language interaction was primarily restricted to pointing gestures and very simple commands, resulting in interaction closer to “verbal tele-operation.” The robot was also missing the ability to recognize semantic entities in the environment, such as rooms and landmarks. Our work is able to handle more complex language and reason explicitly about landmarks and regions in the environment as the robot detects them.

Similarly, the Autonomous City Explorer robot is able to navigate in unknown environments to reach a destination by interacting with pedestrians [15]. This system represents the world as a partial metric and topological environment, and periodically asks for help to reach the destination. The human-robot interaction is limited to simple gestures and a touch screen, both indicating the heading to travel in next (for example, by pointing towards where the robot should go). While the robot asks for help using scripted natural language requests, it does not attempt to understand or follow complex natural language instructions to reach the destination.

Because of the complexity of speech recognition in noisy environments, the language interaction component in these approaches has generally been limited or uni-directional from the robot to people. Despite these limitations, how people give instructions has been studied in many settings, including robot navigation in the TeamTalk corpus [96, 97]. Studies such as these could one day be used for building human-robot dialogue systems.

One such dialogue system was presented by Skubic et al. [146], who developed a multi-modal human-robot control interface that could use speech as one possible input during interaction between a robot and operator. In this work, language was used to command basic actions, provide landmark names (replacing object recognition), and describe spatial relationships to objects. The system could also generate spatial descriptions to describe the location of objects, and perform simple dialogue interactions to learn more about the environment.

While these approaches to human-robot interaction may have only considered very simple language commands, they illustrate the usefulness of complete implemented system and highlight many of the challenges involved with such a complex endeavor. Furthermore, they demonstrate that people are by-and-large willing to interact with complex robots using speech.

2.1.4 Comparison of Problem Space Features

To better compare the problems solved by prior approaches, we list in Table 2.1 the major relevant prior works in understanding natural language for navigation, and highlight the characteristics of the problem these approaches are solving according to two important dimensions. First, the *starting map* may be known or unknown when the robot receives a direction to follow. While assuming an a priori map is reasonable for many applications, in many cases it is impractical or infeasible to assume that a completely-labeled map will be available ahead of time. Furthermore, being able to operate in unknown environments is desirable as it removes any limitation about the environments the approach can operate in.

Second, the constraints placed on the *environment structure* by each approach

Table 2.1: Comparison of domain space features for prior work in natural language direction following. *Starting Map* refers to whether or not the map is known to the robot when it receives a direction to follow. *Environment Structure* refers to any constraints imposed on the structure of the environment (e.g., the approach cannot use landmarks), therefore limiting the types of environments (or commands) each approach can handle. Our approach is the only one that addresses the problem of following directions through unstructured (i.e. real world) environments without an a priori map of the world.

	Problem Characteristics	
Approach	*Starting Map*	*Environment Structure*
Levit and Roy [86]	Known	Open map with landmarks
MAP-TASK [8, 167]	Known	Open map with landmarks
Shimizu and Haas [138]	Known	No landmarks
Kollar, Tellex et al. [73, 158]	Known*	Unstructured
Howard et al. [60]	Known	Unstructured
Kress-Gazit et al. [77]	Known	Constrained, no landmarks
IBL [24, 81]	Unknown	Road intersections only
MARCO [93]	Unknown	Virtual structured†
Chen and Mooney [29]	Unknown	Virtual structured†
Matuszek et al. [99]	Unknown	No landmarks
Our approach	Unknown	Unstructured indoor/outdoor

* Some preliminary results are presented using only local map information built up online.

† These approaches use a game-like simulated environment consisting of grid-aligned intersections and a small set of highly-discriminative landmarks.

also limit the environments that each approach can deal with. For example, some of the approaches only operate in virtual environments with highly regular structure. Other approaches do not reason about landmarks, so would require directions that describe the environment topology only. These constraints further limit the types of environments.

This thesis is the first to address the problem of following directions in unstructured unknown environments.

2.1.5 Summary

Given that language holds the promise of effortless human-robot interaction, it is not surprising that many researchers have focused on utilizing natural language as a medium for interacting with robots or other intelligent agents. In the broad category of understanding natural language directions for robots, the problems addressed so far by the prior literature are following directions in known unstructured environments, and following directions in unknown structured environments. We have shown the limitations of both of these: requiring a complete map of the environment be available is not always feasible, and imposing that the environment obey some structure can be a severe limitation. Ours is the first approach to tackle the problem of following directions in unknown unstructured environments.

We gained several important insights from the works, namely, a world representation, a need to learn the meaning of words directly (free from intermediate formal controllers), and a general framework for understanding directions (sequential decision making). We will use these in our approach described in Chapter 3. Before that, we will continue reviewing related work in other areas that our approach will draw from.

2.2 Imitation Learning

Machine Learning is playing an increasingly important role as robotic systems become more complex. Within this large body of work, techniques that utilize imitation learning (also known as learning from demonstration) are of particular interest, as they benefit from the presence of an expert who can provide examples of the optimal (or desired) behavior [9]. For example, imitation learning has been used to develop a steering control for an autonomous driving vehicle [10, 117], develop helicopter controllers [2, 31, 32], and learn collaborative multi-robot behaviors [30] or task allocation utility functions [40]. In all of these approaches, a person (the expert) provided demonstrations of the desired behavior, and the algorithm used the demonstrations to deduce a way to replicate the same behavior.

One way to learn from human demonstrations is by Inverse Optimal Control [6, 20, 66]. In this setting, the goal is to learn a *cost function* from user demonstrations, such that the demonstrations are optimal under the learned cost function (learning a reward function is an equivalent formulation). In other words, an optimal plan under this cost function should mimic the expert's behavior. The planner can then use this cost function to produce a behavior on new problems. This approach assumes the expert was acting (near) optimally when making decisions, such that the cost function encapsulates the expert's decision making. This formulation is especially useful in settings where an explicit cost function is unknown (or difficult to obtain), and hand-tuning it would be a time-consuming process requiring many iterations of "guess and check." Instead, an expert can demonstrate examples of desired behavior, and the algorithm can use these to learn the cost function. This approach to learning cost functions has successfully been used to learn different driving styles [1], cost functions for arbitrary MDPs [108], route preferences through a city [174], and pedestrian prediction inside buildings [124].

Another way to formulate imitation learning is to reduce it to a problem of structured prediction [156], which is especially well suited for problems where a robot is making a sequence of decisions. In this setting, a structured classifier attempts to predict the expert's action (out of all possible actions). One such approach to imitation learning is Maximum Margin Planning (MMP), presented by Ratliff et al. [121]. It attempts to mimic the expert's behavior and learn a cost function by penalizing disagreements between the expert's demonstrated action and the lowest loss-augmented cost action [121, 122, 123]. This structured prediction formulation has been applied to a variety of problems, including improving overhead imagery interpretation for robot cost maps [123, 139], correctly interpreting perception data for safe navigation over complex terrain [140, 141], and learning driving maneuvers [142]. As an additional benefit, this formulation has an efficient online optimization formulation using the subgradient method and standard convex optimization techniques [121].

While Max Margin Planning and other imitation learning approaches have successfully been applied to many real world problems, recent work by Ross and Bagnell [126] showed that this supervised approach to imitation learning can perform poorly when the training and testing data are not independent and identically distributed. In such settings, the learned policy induces a distribution of visited states that is different than the distribution of states visited by the expert demonstrations. If the learned policy makes a mistake, it will be forced to make a decision from a state it never visited during training, and the errors will compound. To address this issue, Ross et al. [129] reduced imitation learning to no-regret online learning, and presented an iterative imitation learning framework that alternates between training and execution (similar to work on search-based structured prediction by Daumé et al. [35]). This enables the algorithm to collect demonstrations for states that are *induced* by the policy during learning, and results in a policy with performance guarantees over the state distribution it induces [129]. This provided notable performance gains on such diverse problems as autonomous driving, playing Super Mario, handwriting recognition, helicopter control, and image classification [126, 129, 130, 131]. The authors also applied similar ideas to system identification [127], contextual list optimization [132], and cost-sensitive learning problems [128].

2.3 Belief Space Reasoning

Significant uncertainty in a robot's state that is not accounted for when making decisions can result in poor performance. For example, a mobile robot may not be certain about its position, and may run into a wall if it attempts to travel to the goal directly. Instead, the robot can account for the uncertainty in its position by reasoning in *belief space*, resulting in paths that minimize the uncertainty and the time required to reach the destination. While this problem can be formulated as a Partially Observable Markov Decision Process (POMDP), solving POMDPs still remains an intractable problem for most realistic scenarios [63, 88].

In order to plan mobile robot trajectories that minimize the likelihood of becoming lost, Roy et al. [133] developed a model for a map's information content, and then used this model to plan trajectories that take into account the future positional uncertainty. Since completely representing a robot's belief and reasoning about every possible future measurement is computationally intractable, the authors use the simplifying assumptions that the robot's belief can be accurately represented by a Gaussian, and compute the expected localization entropy using only the maximum likelihood sensor measurement. They then compute the environment's information content by combining this expected position entropy with a probabilistic model of dynamic environments (to account for corrupted measurements). This approach then plans trajectories in this information space using a simple graph-based planner, yielding "coastal" paths that navigate to a goal by staying close to walls (a good source of localization information). The resulting paths are slightly longer but reduce the robot's average localization entropy while traveling to the destination. Extending upon that idea, Prentice and Roy [118] applied a belief representation for a probabilistic roadmap planner, creating belief roadmaps that are used to plan informative paths very efficiently. These resulting paths follow similar "coastal" patterns to stay localized.

Beliefs are often represented as a Gaussian. For example, Platt et al. [114] assumes future observations are also normally distributed about the maximum likelihood belief. It then applies conventional optimal control strategies to plan directly in belief space. Similarly, van den Berg et al. [163] represent beliefs by a Gaussian distribution (without assuming maximum likelihood observations) and plans trajectories in belief space for robots equipped with noisy sensors.

For non-Gaussian beliefs, Platt et al. [115] demonstrate an approach for representing samples from the belief space that represent belief hypotheses [115, 116]. This approach generates plans that have a wide margin between the current best hypothesis and all other samples, resulting in actions that are likely to confirm or disprove the current best belief. Various researchers have applied belief space reasoning to understanding the actions of an agent. One approach is to treat action understanding as inverse planning, which provides a computation model for understanding the actions of people [12]. For example, Baker et al. [13] model the desired and beliefs of an agent, and invert a POMDP to infer the best explanation for the observed behavior [11, 13]. Another approach based on the Most Probable Explanation (MPE) assumption is presented by Verma and Rao [165], who formulate the belief space planning problem as a graphical model, where the most likely graphical model is used as an approximation to the maximum a posteriori solution. Using this assumption, the authors formulate planning as the solution to a greedy policy, and show they can learn a policy that infers a goal using partial traces of demonstrations.

Another approach to understanding the actions of an agent is to represent the agent's belief directly and use this belief to generate plans; de Chambrier and Billard [36] learn human search policies from demonstrations in a partially-observable context. This approach represents the person's belief as samples in a particle filter, and reproduces the learned search policy on a robot manipulator to search for a block on a table.

Another method for dealing with uncertainty is to assume that it will be resolved after the next iteration. In this setting, the central idea is to plan by using the expectation over the minimum cost plans, as opposed to the minimum over expected-cost plans. Essentially, this is simplifying the problem by reasoning about the uncertainty for one step into the future, and using minimum cost plans (with no uncertainty) thereafter. QMDP [88] and Hindsight Optimization [172] are two frameworks that utilize this to reason about the uncertainty for a single step.

One successful application of hindsight optimization for robot planning is presented by Kiesel et al. [70], who apply this technique to open world planning problems (where the agent does not initially have complete knowledge about the world state), and demonstrate an efficient solution in a simulated search-and-rescue problem [70] as well as robot planning under temporal uncertainty [69].

When following natural directions in unknown environments, we have several sources of uncertainty: uncertainty about the map, uncertainty about how far the policy is in the direction, and uncertainty in the location of the destination. Uncertainty in the map is the main source of uncertainty we will address, either implicitly (by training a policy to reason about the uncertainty) or explicitly (by representing a distribution over the map). Our approach will make use of the ideas behind Hindsight Optimization to present an efficient solution to the problem of reasoning under uncertainty. Furthermore, we will learn how people reason about the uncertainty in the environment by observing their behavior in a map distribution.

2.4 Active Exploration

Active exploration for simultaneous localization and mapping (SLAM) involves exploration of the unknown parts of the environment while minimizing uncertainty during exploration. This area of research is related to ours in that we begin with an unknown environment and must trade off the cost of gathering information about unknown parts of the world with the cost of executing actions using this knowledge. However, instead of trying to reduce uncertainty in the map or the robot's pose, we are instead trying to correctly follow directions. Our goal is constrained by the task and we may not know for certain when we reach it (as we may never gather a full map). In contrast, SLAM has a very clearly defined goal of gathering a complete map of the environment with no missing pieces. We want to gather just enough information to be reasonably confident that the directions were followed correctly, while minimizing the total distance traveled.

For robot localization, Fox et al. [49] introduced Active Markov Localization for Mobile Robots, where the robot *actively* localizes in a known map by controlling where to move and where to look. This approach computes the information gain (reduction in entropy) of taking any given action, and includes an explicit (hand-tuned) trade-off between the cost of the action and the expected information gain.

In a SLAM setting, similar work by Stachniss et al. [151] attempts to build high quality maps by promoting behaviors like loop closing [150], or using estimates of information gain to trade off the utility of each action by taking into account the expected sensor information gained by the action [151].

Other prior work has studied actively selecting viewpoints [94], trajectories [143], or control policies [71] that minimize errors. Runge et al. [135] use the Expected Value of Information to identify uncertainty that is relevant to an adaptive wildlife management setting.

2.5 Semantic Mapping

Semantic mapping is concerned with enabling robots to build human-centric models of their environment so that they can reason about high-level properties of the environment when interacting with people [80, 109, 119, 173]. Having such rich map representations would enable robots to perform tasks that require reasoning about semantically meaningful components of the map, instead of simple metric navigation tasks. Indeed, understanding natural language directions *requires* a semantic model of the environment so that a robot can reason successfully about the direction (e.g., reasoning about landmarks). For example, the command "go to the kitchen" requires some knowledge of where the kitchen is (or what a kitchen looks like). The two main classes of approaches to building semantic maps are fully autonomous approaches, and approaches driven by human-robot interaction.

Fully-autonomous approaches to building semantic maps do not require human input to generate a map. Prior work has enabled robot to generate topological maps from range data [23], use LIDAR to semantically classify regions in buildings [50, 104, 152], and use 3D point cloud data to semantically label objects [4, 82]. These approaches can be used to generate maps with semantically meaningful properties, such as the locations of corridors, offices, and meeting rooms. Additionally, labeling the location of objects (e.g., computers, chairs, mugs, etc.) in the world can enable robots to reason about these directly (e.g., "bring me the mug") or indirectly (e.g., deduce the location of the office from computers, desks, and chairs). Our work will enable robots to reason within these semantically annotated maps.

Semantic mapping approaches that are driven by interaction with a human operator have the potential to enable robots to build more human-centered representations of their environment, and gather information that would otherwise be hard to deduce autonomously. For instance, recent semantic mapping approaches use a multi-modal interface (including natural language) to build a map of the environment [14, 51, 75, 119]. In these approaches, a user can describe (in natural language) the room the robot is in, or gesture to an object and describe it.

One instance of a user-driven approach to building semantic maps is the Semantic Graph, a framework introduced by Hemachandra et al. [56] as a coupled metric, topological, and semantic map of the environment containing semantically labeled locations described (in natural language) by an operator [56, 168]. This approach extends a standard SLAM metric map with a topological representation of the environment and semantic properties of each region. The semantic properties for detected region labels are inferred using image- and laser-based scene classification, as well as natural language descriptions. During a guided tour to the robot, the user can provide natural language description of the environment the robot is currently in (e.g., “this is the gym”) or parts of the environment the robot does not directly observe (e.g., “the gym is down the hall”). Similarly, Williams et al. [169] use a cognitive architecture to add unvisited locations (described by the user) to a partial map, but only reason about topological relationships to unknown places.

Our work makes use of recent semantic mapping research in two important ways. First, our approach to modeling partially-known environments in Chapter 3 will require access to a semantic mapping component that can reason about metric, topological, and semantic properties of the environment the robot has observed so far. Second, our work on inferring semantic maps from a natural language direction in Chapter 5 will make use of the Semantic Graph [56], but we add the ability to hypothesize *new* locations in the environment using the information contained in the natural language command. This effectively treats language as another sensor that can be used to build a map.

2.6 Summary of Background

Enabling robots to understand natural language directions would be one step towards enabling seamless human-robot interaction. Because of this potential, it is not surprising this interdisciplinary problem has received much attention. The primary difference between the approaches to natural language direction following presented here is whether or not a complete semantic map of the environment is available. Approaches that use a complete map can infer paths using global information from the complete direction and the entire map. Approaches that do not have access to a full map a priori must instead make decisions using only the information about the environment that has been observed. Several previous approaches to this problem operate in constrained structured environments to simplify the problem (for example in simulation or environments without landmarks). In contrast, our approach operates in unstructured environments by learning a policy that can follow directions through unknown environments. In this chapter, we have argued that the problem of following natural language directions through unknown and unstructured environments has not received enough attention.

To tackle this problem, we will leverage major insights gleaned from the prior work.

We will treat following directions as a problem of making a sequence of decisions under uncertainty. This will make use of a policy that predicts an action, and we will train the policy by drawing upon several state of the art imitation learning techniques. This work is related to active exploration in many ways since we start with no map of the world, although in our problem the primary goal is following the direction correctly (not building a map).

We will also treat language as a sensor to generate a distribution of possible maps that extend beyond the robot's sensor range. This makes use of a semantic mapping framework, adding language as a possible input. Since the policy must now reason about a distribution of landmarks, we will additionally bring in work in belief space reasoning. Together, our work will enable robots to follow natural language directions through unstructured unknown environments.

Chapter 3

Following Directions in Unknown Environments

> I never saw a discontented tree.
>
> John Muir

In Chapter 1, we discussed how our particular problem of natural language direction following in unknown environments is one of decision making under uncertainty, where a policy makes a sequence of decisions using only the information (map) it has available. This policy should choose actions to explore the environment (building up the robot's knowledge of the world), backtrack if the robot made a mistake, and explicitly declare when the robot has reached the destination.

In the prior work presented in Chapter 2, we analyzed various approaches to similar problems, and gleaned several important lessons. Among these, we gain a way of representing the world (using a combined metric, topological, and semantic map), a general approach of learning the meaning of actions directly (free from intermediate formal controllers), and a representation of actions in the world that uses features. Additionally, we know that our policy must have several properties: it should reason about partial information and make decisions under uncertainty (e.g., even in the absence of landmarks), it should recover from mistakes it makes during execution, and it should make a sequence of decisions until it explicitly declares that the robot has reached the destination.

In this chapter, we further detail our approach, specifically implementing a system that enables robots to follow natural language directions through unstructured and unknown environments. At the heart of the policy lies the equation we introduced in Section 1.2:

$$\pi = \underset{\text{actions}_t}{\operatorname{argmin}}\ \text{cost}\left(\text{position}, \text{action} \mid \text{language}, \text{map}_t\right). \tag{3.1}$$

Looking more closely at the components of this equation, we identify the key steps required to formalize the policy. First, we need to model the information contained in the natural language command given by the user (the language). Second, the robot needs to build up a partial model of the environment, using a combined semantic-topological-metric representation (the map). Third, the policy needs to enumerate the set of actions available at any time, so that they can be evaluated under the cost function (the action set). Finally, given an action, command, and partial world, the policy must evaluate the cost of that particular action. The policy will do this using a feature representation of the action. We will describe each of these in this chapter, as well as the entire algorithm for direction following in unknown environments. We leave until Chapter 4 a discussion of how the policy is *learned*, and for now assume that the cost function is given.

For conciseness, Table 3.1 defines the following symbols to help represent the policy formulation more succinctly. We can now rewrite Equation (3.1) as:

$$\pi\left(x\right) = \underset{a \in A_t}{\operatorname{argmin}}\, c\left(x, a \mid \Lambda, S_t\right). \tag{3.2}$$

Table 3.1: Symbols used in policy formulation.

t	current time
x	current position
Λ	natural language command
S_t	current partial map at time t
A_t	set of possible actions available, given the map S_t
a	a single action
c	the cost function

For notational convenience, we introduce a state variable s that encapsulates the current pose, the map, and the language command:

$$s := \{t, x, S_t, \Lambda\}. \tag{3.3}$$

This enables us to further simplify the policy representation as a cost function evaluated over state-action pairs:

$$\pi\left(s\right) = \underset{a \in A_t}{\operatorname{argmin}}\, c\left(s, a\right). \tag{3.4}$$

In this setting, the policy enumerates a set of actions A_t available from the current state, and chooses a single action to execute on the robot. More concretely, each individual action will either take the robot to a new place in the environment (exploring new parts of the map), backtrack to a previously visited location (if the robot made a mistake), or explicitly declare that the robot has finished following the direction. The policy in Equation (3.4) considers the cost of each possible action, and selects the one with the lowest cost. The goal is for the robot to choose the right set of actions A_t and the correct cost function. In this dissertation we assume that the cost function takes the form of a linear combination over features ϕ of the state-action pair:

$$c\left(s, a\right) := w^T \phi\left(s, a\right) \tag{3.5}$$

While many other cost functions are possible, a linear cost function is efficient to compute and learn.

The remainder of this chapter describes our technical approach to enabling robots to follow natural language directions through unknown environments. We first describe our model for the language command Λ, which is split into a sequence of semantic clauses (Section 3.1). We then model the partially-known environments with a semantic map S_t that is built up online (Section 3.2). In addition to the metric and topological structure of the environment, this map contains semantic information of places and objects. We then describe in Section 3.3 the detailed representation of the policy, including the state and action spaces, and the complete algorithm for following directions in unstructured unknown environments. Finally, we describe the features used in the policy's cost function in Section 3.4.

3.1 Modeling Spatial Language

Natural language is used extensively when people collaborate, and would provide a similarly natural and flexible way of interacting with autonomous robots. Such interaction would not require specialized interfaces, extensive user training, or knowledge of robotics. However, several factors make following natural language directions challenging for robots. First among these, language that is only constrained by the task may engender a large vocabulary of words that users may choose from [157]. In other words, there are many ways of expressing directions that map to the same action, and users may direct robots towards the same intended destination with very different commands. Additionally, untrained users may have different spatial representations of the world [72]. These challenges are due to the complexity of natural language communication; enabling robots to understand language will require dealing with this complexity.

Fortunately, most spatial language directions exhibit properties that we can leverage to simplify this challenging problem. This linguistic structure was described as a cognitive concept by Jackendoff [62] and colleagues [83, 155], and was later formalized as a computational concept by Tellex [157] and Kollar [72] in the form of Spatial Description Clauses (SDCs). SDCs have been successfully used in many prior approaches of robot natural language understanding, particularly following directions and robot manipulation [41, 61, 72, 73, 74, 144, 158].

In this work, we will primarily make use of the following two key properties of natural language direction. First, natural language directions are *sequential*: each clause in a direction refers to one step along the path, ordered from the start to the destination [72]. This enables us to decompose one long direction into several shorter clauses. Second, each clause has *structure*: parts of the direction refer to different meaningful terms; verbs prescribe what to do and where to go, spatial relations describe the relative geometry between the path and the landmark, and landmark references describe what objects will be visible from the path during navigation [73]. This enable us to reason about semantically meaningful components of language, instead of attempting to learn a model for all natural language sentences. For example, we will be able to learn models for individual verbs (such as "turn right") that are independent of the landmark. These two properties of spatial language directions

Table 3.2: Sequence of SDCs for the sentence "Turn right towards the elevator and go through the doors." The command is decomposed into two SDCs (Λ_0 and Λ_1), each with semantically meaningful components.

	Verb	**Landmark**	**Spatial Relation**
Λ_0	turn right	the elevators	towards
Λ_1	go	the doors	through

are encapsulated in a representation known as *Spatial Description Clauses* (SDCs), introduced by Tellex [157].

Described more formally, the SDC representation enables us to represent a complex natural language direction Λ as a sequence of Spatial Description Clauses:

$$\Lambda \rightarrow [\Lambda_0, \ldots, \Lambda_N], \tag{3.6}$$

where each clause contains some structure. This enables us to simplify the problem of following a complete direction into several sub-problems, each following a single clause in the direction. Moreover, each SDC Λ_i consists of several components:

- *Verb:* an action to take.
- *Landmark:* an object described in the command.
- *Spatial Relation:* a desired geometric relation between the landmark and the path.

Any of these fields can be unlexicalized and therefore only specified implicitly. This semantic structure provides a decomposition of language into components, each of which will (as we will later show) be modeled separately. A single SDC models the actions necessary to follow a short command, and a sequence of SDCs forms a complete command.

Given a natural language direction (for example, the one shown in Table 3.2), we first decompose the command into a sequence of two SDCs that the policy will complete in order. Then, we extract the semantically meaningful components of the language (verbs, landmarks, spatial relations) from each SDC. This formulation provide a semantically meaningful representation of the intent of the natural language direction, while remaining flexible to the many possible ways of expressing the same thing (by extracting the key components in the command). Furthermore, it does not rely on formal controllers that directly prescribe how the robot should act.

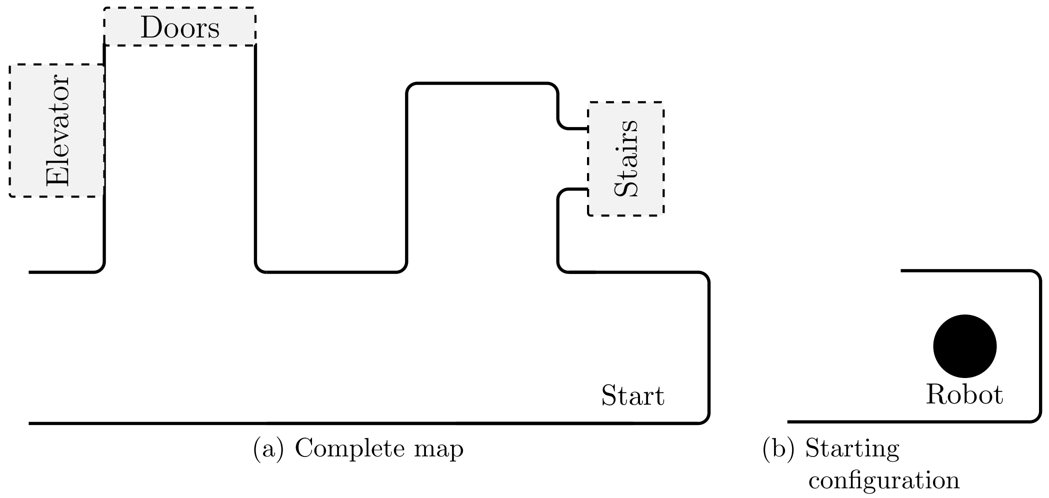
Although Spatial Description Clauses will enable us to understand natural language directions by reasoning about its components, there are some directions where this SDC formulation does not hold. In particular, some directions are best represented as a hierarchy of SDCs, or may require reasoning about complex conditionals: for example negation (do not go through the doors) or counting (the third door on the right). Nonetheless, in spite of these limitations, this formulation enables tractable inference for following directions, since we can treat the problem as a sequence of SDC-following procedures. Given our assumptions about the sequential and decompositional nature of natural language directions, this simple model is expressive enough to capture much of the complex information contained in natural language.

3.2 Modeling Partially Known Environments

The previous section described our representation of the information contained in the natural language command as a sequence of semantically structured clauses. We now discuss the representation of *world*: a semantic map that evolves over time as the robot travels in the environment. This semantic map S_t contains three main components, outlined in Table 3.3: a metric layer, a topological layer, and a set of semantically labeled objects. We make no assumptions about the structure of the environment or how the robot builds up the map, only that it can represent and update the metric, topological, and semantic map layers. This map provides a human-centric model of the environment that is interpretable by both people and robots, and the policy will use it to reason when following directions. We now describe in more detail each component of the partial semantic map, using a running example of a robot operating in the environment shown in Figure 3.1. In this hypothetical explanatory scenario, the robot's sensing range is limited, and the known information at the start (Figure 3.1b) is a very small subset of the complete map.

Table 3.3: Components of the semantic map S_t at time t.

X_t	metrical representation of the free and occupied space in the environment
G_t	topological representation of how regions are connected
$\mathcal{O}_t$	a set of semantically labeled objects and regions



(a) Complete map

(b) Starting configuration

Figure 3.1: Illustrative scenario we use as a running example to describe our approach: complete map of the environment (3.1a) and the resulting partial map for the starting configuration (3.1b). Whereas we show all landmarks in the complete map, the robot must make decisions using only the partial map information.

Metric Layer

The primary purpose of the metric layer X_t is to ensure the robot can navigate in the environment without colliding with obstacles. We do not impose restrictions on how this metric map is generated, only that it exists. For example, it can be generated by a Simultaneous Localization and Mapping (SLAM) module running on the robot. Similarly, we assume a motion planner exists that will be able to navigate the robot to a chosen destination as long as a collision-free path exists. The details of the planner are not important. For instance, in this dissertation we have applied our approach on a variety of motion planners running on robots, including planners that are sampling-based [67], graph-based [17, 18, 33, 164], and grid-based [46, 47, 153].

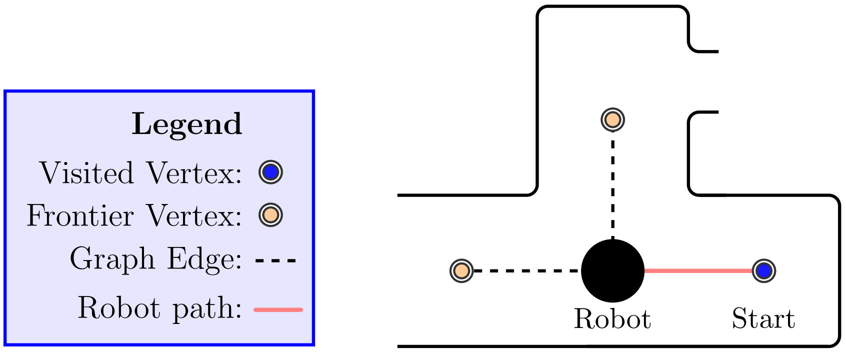


Figure 3.2: The topological layer of the map stores where the robot has been and where it could go. As the robot moves in the environment (—), it builds up a topological graph. This graph consists of places the robot has already been (●), and places the robot could go to gain information (◎).

Topological Layer

The topological representation of the world provides a way to reason over paths in the environment. As the robot travels, it builds up a graph representation of the places it has detected:

$$G_t = \{V_t, E_t\}, \tag{3.7}$$

where the vertices $v \in V$ represent possible viewpoints that are connected by edges $e \in E$ representing allowable robot travel segments. Since this graph is generated incrementally, the robot adds two types of nodes to this graph: nodes that represent *previously visited* locations $\mathcal{V}$, and nodes that represent *frontiers* $\mathcal{F}$. Frontier nodes lie between explored space and unexplored space, and are inspired by work on frontier-based exploration [171] and other active exploration methods [94, 151]. Edges connect any two nodes that have a feasible path between them. As the robot moves through the environment towards unknown areas (i.e., towards frontier nodes), its sensors can observe into the unexplored space. This "pushes forward" the boundary of knowledge, and adds new nodes to the graph (both previously visited and frontier nodes). For example, in Figure 3.2 the robot adds two frontier nodes to the graph G_t. The robots removes frontier nodes once it reaches their location (they are now a visited node).

This topological representation is beneficial in two ways. First, the vertices

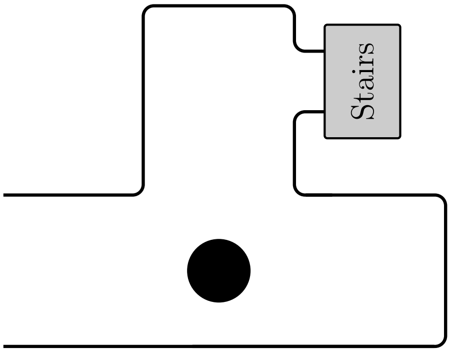


Figure 3.3: The semantic layer of the map is a set of semantically annotated objects. When the robot's perception detects an object in the environment such as the stairs (▭), it adds it to the set of known objects $\mathcal{O}_t$.

connected by feasible travel edges gives the robot the ability to generate and reason about partial paths in the environment. Second, the frontier nodes are an explicit representation of the information it does not yet have, and the policy will use this to bias actions towards exploration. These will become core components of our approach to following natural language directions in unknown environments.

Semantic Layer

The final component of the semantic map S_t is the set of all semantically annotated objects $\mathcal{O}_t$ detected by the robot as it travels in the environment. The objects can be physical things such as a water fountain or a door, landmarks such as a staircase or an elevator, or a region in the world such as a kitchen or an intersection. Each object o has a semantic label o_{tag} and some geometry (represented as a sequence of points) $o_{\text{points}} = [p_0, \ldots, p_n]$. An object can be used for navigation after it has been detected. In other words, as the robot is following a direction the policy can only use landmarks already in $\mathcal{O}_t$ to reason about. For example, in Figure 3.3 the map only contains the stairs object (the elevator and doors are not yet known).

Summary

The semantic map S_t forms the robot's world knowledge during execution. it consists of the metric, topological, and semantic layers presented above, and can be written as:

$$S_t = \{X_t, G_t, \mathcal{O}_t\}. \tag{3.8}$$

This semantic map is generated incrementally during robot execution, and will be used throughout the rest of this work to reason about the possible actions, any objects in the world, and how these relate to the command represented as a sequence of Spatial Description Clauses. Note that we have made no assumptions as to the structure of the world: as long as the robot is able to detect objects and represent valid paths in the environment we can apply our approach for following natural language directions in an unknown environment.

3.3 Policy Representation

With the above representations for the command and map, we detail the policy π that maps from states to actions by minimizing the cost function over a set of possible actions:

$$\pi\left(s\right) = \operatorname*{argmin}_{a \in A_t} c\left(s, a\right) \tag{3.9}$$

$$= \operatorname*{argmin}_{a \in A_t} w^T \phi\left(s, a\right). \tag{3.10}$$

The state s consists of the current time t, position x, map S_t, and command Λ (Equation (3.3)). In this section we describe how the policy's state evolves as the robot moves through the environment, and how to compute the set of actions A_t given the current state. Our choice of action set is highly dependent on the semantic map structure we have previously described. The set of actions encapsulates the three desired types of behavior: exploration to new places, backtracking to somewhere the robot has already been, and explicitly declaring when the policy has finished following the direction. We also detail the entire algorithm for following directions in unknown environments.

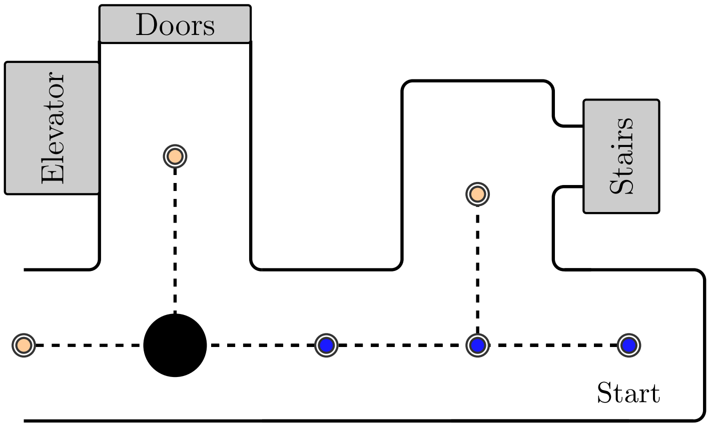


Figure 3.4: The policy state updates as the robot travels through the environment. Here we show the map update: the topological graph contains more nodes (both visited and frontiers), and the set of semantically labeled objects now includes the elevator and doors.

State

The state represents the information necessary and sufficient for the policy to make a sequence of decision. This state evolves over time as the robot travels through the previously unknowns parts of the environment. The components of the state are the time, the robot's position, the current map, and the input direction.

The state initialization is straightforward. Since the robot does not have an a priori map of the environment we set its starting position at the origin (v_0) and add this vertex to the topological graph in the map (the map is empty otherwise). The direction Λ is broken down into a sequence of SDCs $[\Lambda_0, \ldots, \Lambda_N]$, as described in Section 3.1.

During execution, the state updates to reflect the latest information available to the robot. The time increments and the pose x updates to reflect the robot's current position. The three layers of the map also update in the following manner. The metric map reflects the latest sensor information available to the robot. The topological graph adds both visited and frontier graph nodes (to $\mathcal{V}$ and $\mathcal{F}$ respectively), as well as connects valid travel segments between nodes by edges. After the robot completes a travel segment (i.e. action), it adds a visited node at that position. The robot also adds frontier nodes at locations known to be empty but that have not been visited, up to some maximum distance. Lastly, as the robot detects objects with its perception

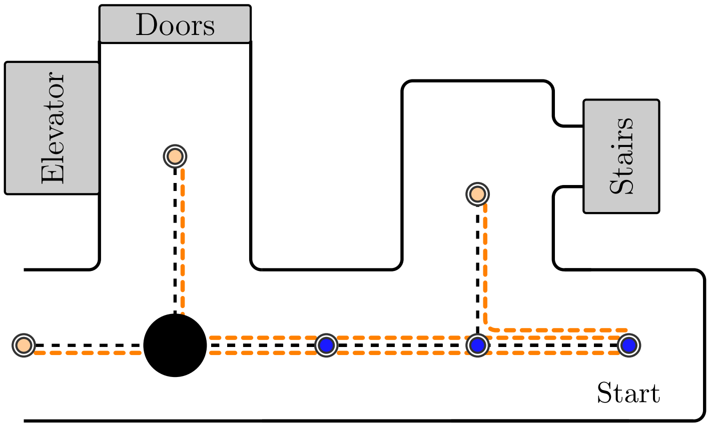


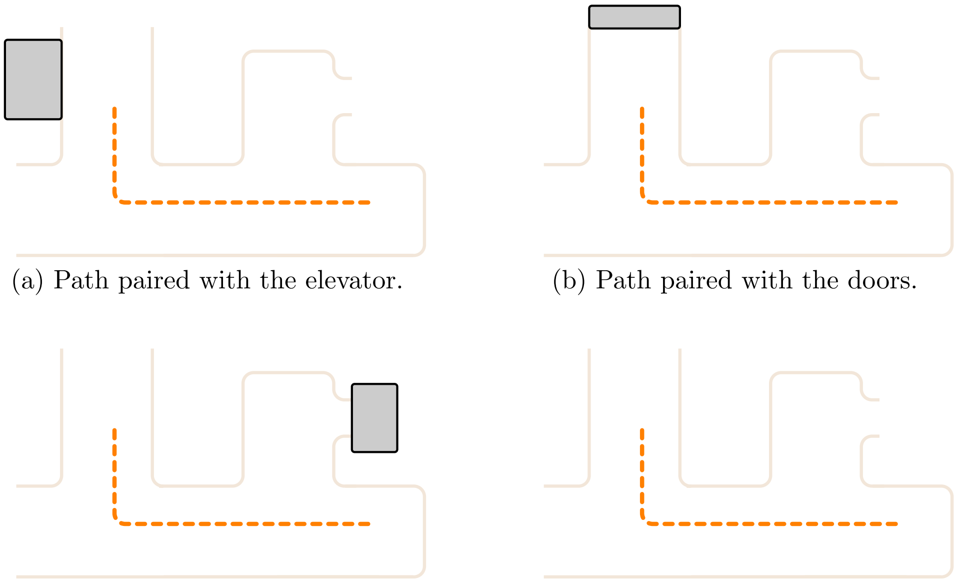
Figure 3.5: Actions are places the robot can go next. Each action is a path in the topological graph (---) paired with a landmark. For simplicity, we only show the three actions that end at frontier nodes, and do not show the landmark associated with each path in this figure. Additionally, the stop action (not shown) is always available to the robot.

system, it adds them to the set of semantically labeled objects. This updated state, illustrated in Figure 3.4, will be used to compute the set of valid actions.

Action

Given a state s, the policy must compute a set of actions A_t that represent all allowable actions the robot can take at time t. Like the state, the action set evolves over time, since the policy computes it using the semantic map S_t and the current position x. Each action is a path from x to a vertex $v \in V_t$ paired with an object $o \in \mathcal{O}_t$. The object may be undefined, to account for objects that are not yet visible or SDCs that do not specify one. A separate action a_{stop} declares that the robot has reached the SDC's destination: the policy moves to the next SDC if one exists (otherwise the robot has finished following the entire direction). Figure 3.5 shows several feasible actions that end at frontier nodes in the topological graph for the updated policy state.

Each action can either explore (if the path ends at a frontier node in $\mathcal{F}$), backtrack (if the path ends at a previously visited node in $\mathcal{V}$), or stop (if the action is a_{stop}). While the policy reasons about these types of actions when computing features of various state-action pairs, we do not explicitly limit the set of available actions



(a) Path paired with the elevator.

(b) Path paired with the doors.

(c) Path paired with the stairs.

(d) Path paired with no landmark.

Figure 3.6: Since each action is a path paired with a landmark, the policy is searching over the space of possible groundings for the command. Here we show four different actions that use the same path: using previously detected landmarks (figs. 3.6a to 3.6c) or no landmark (fig. 3.6d). This is a subset of the valid policy actions since there are other paths in the environment.

or otherwise reason separately about the different classes of actions (exploration vs. backtracking vs. stopping). The set of actions available to the policy is formally represented as:

$$A_t = \{\text{path}\,(x, v) \in G_t \cup a_{\text{stop}}\} \times \mathcal{O}_t, \tag{3.11}$$

where path (x, v) is a valid path in the graph from x to a vertex $v \in V_t$. This set of actions represents the possible *groundings* of the natural language command: the objects o are possible groundings for the landmark field of the SDC, and the paths are possible groundings for the action (i.e. the verb and spatial relation fields). As such, each action a is one possible grounding for the Spatial Description Clause, and the policy will search over these groundings. In other words, the action set will include the same path paired with all known landmarks (see Figure 3.6).

Intuitively, an action represents one step along the direction's full path. The policy will make a sequence of decisions, choosing a single action to execute from the

Algorithm 1 Following natural language directions through unknown environments.

```
procedure FOLLOW DIR($\pi, \Lambda$)
    $t = 0$
    $x_t = v_0$
    $X_t = \emptyset$                                        ▷ No metric map
    $G_t = \{V = \{v_0\}, E = \emptyset\}$                    ▷ Unknown environment
    $\mathcal{O}_t = \emptyset$                               ▷ No known objects
    $S_t \leftarrow \{X_t, G_t, \mathcal{O}_t\}$
    $\Lambda \rightarrow [\Lambda_0, \ldots, \Lambda_N]$      ▷ Extract sequence of SDCs from $\Lambda$
    cur_sdc = 0                                               ▷ Begin with the first SDC
    repeat
        $t \leftarrow t + 1$
        ▷ Update semantic map with sensor measurements $z_t$
        $S_t \leftarrow$ PERCEIVE_WORLD($z_t$)
        $s \leftarrow \{t, x, S_t, \Lambda_{\text{cur\_sdc}}\}$  ▷ Update state
        $A_t = \{\text{path}\,(x, v) \in G_t \cup a_{\text{stop}}\} \times \mathcal{O}_t$   ▷ Compute the action set
        $a \leftarrow \pi\,(s)$                               ▷ Choose a single action
        $x_t \leftarrow$ MOVE(a)
        if $a == a_{\text{stop}}$ then
            cur_sdc ← cur_sdc + 1                             ▷ Move to next SDC
        end if
    until cur_sdc > $N$
    return
end procedure
```

(evolving) action set (see Figure 3.7). For example, if the robot were traveling down a hallway, the action set would include a single frontier node further down the hallway. When the robot reaches that node, another action would become available (traveling a little bit further still). Additionally, since the action set still contains potential actions in other areas of the environment (e.g., down a different hallway), the robot is free to backtrack at any time and go into a different part of the environment. Lastly, the policy must explicitly declare when the robot has finished following the direction by calling the a_{stop} action. This action will complete the current SDC Λ_i, and move to the next one (Λ_{i+1}) if it exists, or return.

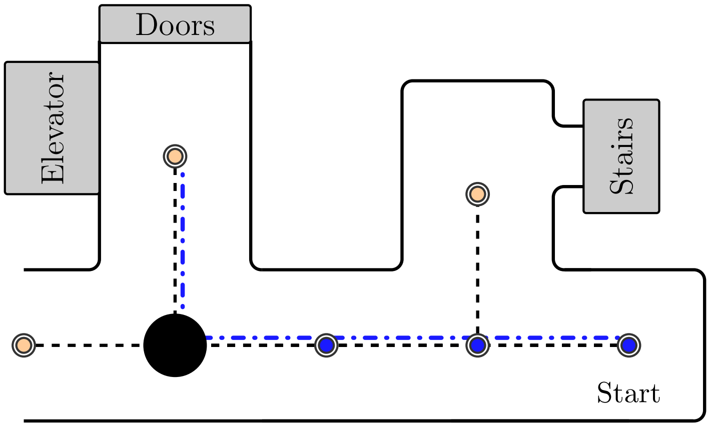


Figure 3.7: The policy chooses a single action to execute on the robot (·--·). The robot will then move to the endpoint of the action (in this case, towards the elevator).

Policy

The policy evaluates all actions $a \in A_t$ available from a given state s, and chooses the single action that minimizes the cost of the state-action pair in Equation (3.9). We will defer a discussion on learning the cost function until Chapter 4.

The complete algorithm, described in Algorithm 1, shows the entire procedure for following natural language directions through unknown environments. We initialize the semantic map (Line 7) with an empty metric map, a topological graph containing only the starting vertex v_0, and no semantically labeled objects. We also extract a sequence of Spatial Description Clauses from the command Λ (Line 8).

During execution, the robot perceives the world (Line 13). This updates the map (as described above) by updating the metric map (with any free/occupied space), the topological graph (with any new frontier nodes in free space and visited nodes), and the set of semantically labeled objects (with any new object detections). Then the algorithm enumerates the set of available actions, the policy evaluates the respective cost of each action, and returns the minimum cost one (Line 16 corresponds to the action shown in Figure 3.7). The robot executes this action to move to a new place in the world, unless the policy selected the stop action. Throughout the procedure the algorithm keeps track of the current SDC, and transitions when a stop action is called until there are no more SDCs in the sequence.

Summary

The policy described here is at the heart of our approach to following natural language directions in unknown environments. It makes use of several important components in natural language processing and semantic mapping, and ties them together in a simple and elegant algorithm. The policy reasons about the world known so far through a semantic map S_t, and enumerates a set of actions A_t representing the various places the robot could go next. The policy updates its state as the robot gains new information, representing the increasing knowledge about the world. Because our map representation is quite flexible and reflects what most robots would already be equipped with, we are able to reason about actions in complex unstructured environments. The policy's cost function is a weighted sum of features, and we will now describe how the features of a state-action pair are computed.

3.4 Feature Representation

To evaluate the cost of action a from state s, the policy must compute a linear combination of features of the state-action pair (Equation (3.10)). The features are a vector of values that describe (numerically) the properties of each action, and the goal of the policy is to map from those features pairs to a cost that relate *how well* the given action follows the direction: a low cost for an action would mean the action follows the direction well, a high cost would mean it does not. This feature representation provides two key benefits:

- features relate the natural language command to the action under consideration, and
- features enable the policy to generalize to new directions and even different environments.

In this work, we use a combination of spatial and linguistic features, primarily adapted from work by Tellex [157] and Kollar [72]. These features fall into several categories that we will describe below. The inputs required to compute features are the geometry of the path represented by the action, the object associated with the action, the SDC fields for the current SDC, and the topological graph stored inside the semantic map.

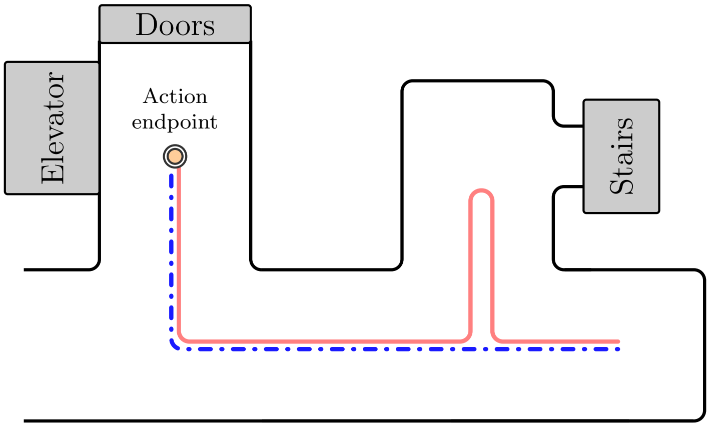


Figure 3.8: When computing features for an action, we only consider the shortest path between the start and the action's endpoint (·-·-·). We do not consider the entire trajectory history (—), as this would result in features that are not meaningful with respect to the given direction. In this example, they would not be representative of a "right turn".

It is important to note that for a given action $a \in A_t$, the policy computes features for the shortest path between the start of the current SDC and the action's endpoint. This path does not take into account any previous (potentially backtracking) actions, which ensures previous actions do not obscure the meaningful features of the action we wish to evaluate. For example if the robot turns into the wrong hallway, backtracks, and then finally takes the correct right turn (shown in Figure 3.8), we compute features of the final right turn (without backtracking). Otherwise, features computed over the entire history (including backtracking) would lose their meaning with respect to the direction the policy is following. Though the extra distance is not desirable in the long run, we are interested in features of the final complete path to the action's endpoint, without taking into account the backtracking when computing features. If action features were computed over the entire action history (including backtracking) the distribution of features for the same direction would be vastly different depending on the history, which is undesirable.

Geometric features of the action

Geometric features are a function of the path and object geometry: they describe the shape of the path, the geometry of the object, and the relationship between the two. These are useful to relate the action to the *verb* and *spatial relation* fields of the Spatial Description Clause (in Section 3.1).

Some example features in this category are: the total angular change of the path (this might be correlated with actions that turn or go straight), the area occupied by the object, and the displacement of the path start and end with the landmark's center of mass (this might be correlated with actions that go towards or away from the landmark).

Note that we compute all spatial features for any particular action, regardless of the contents of the SDC fields (in other words, we do not select which features to compute). Our learning algorithm will determine which features are important for which terms. Additionally, the detected object's label (name) is not used here: this enables us to learn models for verbs that generalize across different object labels: a path that turns right to an elevator will have similar geometric features to a path that turns rights to the stairs. This enables us to decompose reasoning about the shape of the action and the semantic label of the object in the world, without requiring examples of directions with every combination of path and landmark (for instance, turning right to every possible object). These features are a key part of grounding the verb and spatial relation fields in the Spatial Description Clause: the policy can learn a mapping from words to geometric features.

Linguistic features of the action

Linguistic features are a function of the object name and SDC landmark field, and express a similarity metric between what the direction *says* and what the robot *sees*. These are useful to describe the *landmark* field of the Spatial Description Clause, and to generalize across synonyms or semantically similar references to the same object (for example, "elevators" and "lift"), as well as objects that are likely to co-occur near each other (for example, a refrigerator is likely to be in a kitchen). This ability to reason about co-occurrences is especially important for robots, since perception systems will never be able to detect every possible landmark, and some may be very challenging (for example, detecting kitchens). However, by reasoning about the context of objects that are likely to co-occur near each other, the robot can follow directions that refer to undetectable objects by instead perceiving other objects in the environment.

We expand upon a system developed by Kollar [72] that utilizes WordNet [45, 101] (a lexical database of English words) and a database of tagged images (extracted from Flickr) to compare the label of objects in the world and the SDC's landmark field. The inputs are pairs of strings, and the output is a vector of real values indicating how similar the two words are according to various metrics in these two systems. These linguistic features enable the policy to reason about the similarity between the landmark mentioned in the direction and objects in the world the robot can detect. As such, these features are a key part of grounding the landmark field in the Spatial Description Clause: as the policy considers all possible actions (with different objects, see Figure 3.6), it can consider the similarity between the object used for the action (Equation (3.11)) and the landmark described in the SDC.

Features of the stop action

Features for the stop action enable the policy to reason explicitly about the stop action a_{stop}, which either transitions to the next SDC or exits the direction following procedure. We first compute an indicator feature for whether or not the current action is a stop action. Additionally, if the action is a stop action we compare the geometric features of the action with the average geometric features of previously observed paths for the same verb. For example, if the command is to "turn right," the policy will compare the geometric features of the stop action with the average of previously observed paths for right turns. This effectively measures the distance between features of the action under consideration and a canonical path, and enables us to change the cost for actions depending on whether or not the path matches the expected features for this action. In other words, we can measure how similar the current action is to other actions (right turns, left turns, etc.) computed from a corpus of complete paths. These features help ensure that the complete path under the stop action is representative of the entire direction if the policy is stopping.

Combinations of features

Combinations of the above features enable us to reason about path shapes and landmark names that occur at the same time. For example, we use the Cartesian product of the geometric and linguistic features to represent the combination of the path shape and object label. This provides a more expressive feature space, by reasoning about actions that match both the expected shape and the SDC's landmark (for example, turning right *and* seeing stairs) instead of either the shape or landmark name.

3.5 Chapter Summary

In this chapter we have shown how to formulate the problem of following natural language directions in unknown environments as one of decision making under uncertainty. Under this formulation, a policy (π) makes a sequence of decisions (actions) that follows a command, using its knowledge about the world so far. We leverage prior work in Natural Language understanding to extract structure inherent in spatial language directions and to model a direction as a sequence of semantically annotated clauses. We also model the partially-known environment using a mixed metric, topological, and semantic map (S_t) that contains a model of the free and occupied space around the robot, and the location of places in the environment the robot has either previously explored or that lie on the edge of explored space (frontiers). The map also stores the set of all landmarks that the robot has previously observed, so that the policy can reason about their geometry and semantic labels. The policy state (s) consists of the map, the language command, and the current position of the robot in the map.

As the robot moves in the environment, our approach incorporates new information about the world to update the policy's state. Given a particular state, we can then compute the set of possible actions available to the robot. These actions encompass exploration (to new parts of the environment), backtracking (to previously visited parts of the environment), and declaring that the current clause is finished (when the policy believes the robot has completed following the direction). After the policy enumerates all possible actions, it commands the robot to execute the one with the lowest cost. As the robot moves, it receives new observations that will provide the policy with more information to make the next decision. The entire process repeats until the policy calls the stop action, explicitly declaring that it has completed following the direction.

The policy makes use of a cost function (c) that is parameterized as a linear combination of features of the state-action pair ($\phi(s, a)$). These features are a numerical (vector) representation of the action, and are used to relate the action and command. This makes the cost function efficient to compute and learn. Learning the policy will be the focus of the next chapter.

Chapter 4

Imitation Learning in Unknown Environments

> Success is the ability to go from failure to failure without losing your enthusiasm.
>
> Winston Churchill

The previous chapter described our approach to following natural language directions through initially unknown environments. We leveraged several useful properties of spatial language directions to break down the problem into understanding (and following) a *sequence* of structured clauses that contain semantically annotated information about what the robot should do and see along the way. We introduced a representation for partially-known environments that combines metric, topological, and semantic map information. This in turn enables the policy to compute a set of feasible actions available to the robot, namely, places the robot can go next. The policy then evaluates the *cost* of each action, and commands the robot to execute the minimum cost action. The cost function is parameterized as a linear combination of features, several of which we described. In this chapter, we focus on learning the weights w that are at the heart of the policy's decision making.

We train the policy through *imitation learning*, using demonstrations of people (henceforth referred to as experts) giving and following directions. More specifically, we use inverse reinforcement learning to learn the cost function c the expert was possibly optimizing when providing demonstrations. Intuitively, the goal is to maximize the agreements between our policy and the expert demonstrations, so that the policy predicts the expert's action out of all possible actions. We can treat this as an iterative imitation learning problem, where the learning algorithm iteratively updates the cost function by comparing the decisions made by the policy and the expert demonstration.

The demonstrations consist of a natural language direction, its associated path, and the landmarks used to follow that direction. Each direction in the training set has an associated parsed Spatial Description Clause (SDC, Section 3.1) structure, and the complete path is chunked so that each segment maps to the relevant SDC in the direction. The landmark used for following each SDC is also annotated by the user.

In this chapter, we first show in Section 4.1 how to formulate this learning problem as multi-class online learning, and define the loss function that our learning algorithm will optimize to maximize agreements between the policy and the demonstrations. In Section 4.2 we further explore learning in unknown environments, so that the policy reasons about uncertainty. We then discuss in Section 4.3 an application of a novel imitation learning framework to our problem that will enable the policy to learn to recover from mistakes by training on states that include failures. In Section 4.4 we then show how to use the demonstration data to compute a demonstration policy given a path. Finally, we demonstrate in Section 4.5 this approach on a corpus of natural language directions through a floor of MIT's Stata center.

4.1 Formulation as Online Learning

We treat action prediction as a multi-class classification problem, where we want the policy to predict the expert's action out of all actions available from a given state. Given traces of people following directions, we learn the policy for single-SDC segments. We assume that the expert's policy π^* minimizes the unknown immediate cost $C\left(s, a^*\right)$ of performing action a^* from state s:

$$a^* = \pi^*\left(s\right) := \operatorname*{argmin}_{a \in A_t} C\left(s, a\right). \tag{4.1}$$

Since we do not directly observe the true costs of the expert's policy, we must instead minimize a surrogate loss function that will penalize disagreements between the chosen action $\pi(s)$ and the expert's demonstrated action $\pi^*(s)$ over all examples:

$$\ell(s, \pi^*, \pi) \propto \sum_{\text{examples}} \pi(s) \neq \pi^*(s). \tag{4.2}$$

Intuitively, if our policy π always agrees with the expert's policy π^* for any state s, then the policy will follow the direction correctly.

The policy selects an action that is different from the expert's action ($a \neq a^*$) if and only if the cost of that action ($w^T \phi(s, a)$) is lower than the cost of the expert's action ($w^T \phi(s, a^*)$). This enables us to treat direction following as a multi-class structured prediction problem, and penalize disagreements between our policy and the expert's using the multi-class hinge loss [34]:

$$\ell(s, a^*, w) = \max\left(0,\ 1 + w^T \phi(s, a^*) - \min_{a \neq a^*}\left[w^T \phi(s, a)\right]\right). \tag{4.3}$$

Intuitively, the loss in Equation 4.3 is zero when the cost of the expert's action is lower than the cost of all other actions by a margin of one. If the cost of the expert's action is more than the cost of another action (again by a margin), the loss is positive and follows the well-known multiclass SVM loss. This loss can be rewritten as:

$$\ell(s, a^*, w) = w^T \phi(s, a^*) - \min_{a}\left[w^T \phi(s, a) - l_{sa}\right], \tag{4.4}$$

where the margin $l_{sa} = 0$ if $a = a^*$ and 1 otherwise. This loss augmentation term ensures that the expert's action is better than all other actions by a margin [121]. Adding a regularization term λ to (4.4) yields our complete optimization loss:

$$\ell(s, a^*, w) = \frac{\lambda}{2}\|w\|^2 + w^T \phi(s, a^*) - \min_{a \in A_t}\left[w^T \phi(s, a) - l_{sa}\right]. \tag{4.5}$$

Although this loss function is convex, it is not differentiable. However, we can optimize it efficiently by computing the subgradient of Equation 4.5 and computing action predictions for the loss-augmented policy [121]:

$$\frac{\partial \ell}{\partial w} = \lambda w + \phi\left(s, a^*\right) - \phi\left(s, a'\right), \tag{4.6}$$

where a' is the best loss-augmented action from state s:

$$a' = \operatorname*{argmin}_{a \in A_t} \left[w^T \phi\left(s, a\right) - l_{sa}\right]. \tag{4.7}$$

The subgradient leads to the following update rule for w:

$$w_{t+1} \leftarrow w_t - \alpha_t \, \frac{\partial \ell}{\partial w_t}, \tag{4.8}$$

with a learning rate $\alpha_t \sim O\left(1/t\right)$. We repeat this update for T time steps, or until the weights converge.

Intuitively, the update rule decreases the cost associated with features of the expert's action a^*, and increases the cost associated with features of the predicted action a'. If the expert action matches the policy's, the gradient will be zero and the policy weights will not change (ignoring regularization), as we desire.

4.2 Training in Unknown Environments

Our aim is to have a policy that reasons about the uncertainty inherent in partially-known environments. As we discussed in Chapter 2, some prior approaches train a model on a full world map and then attempt to apply it to a partial world map. Our approach instead trains the policy directly in unknown environments, and treats learning as several iterations of applying the policy to the training data and then updating the policy based on the expert demonstrations.

To do this, we embed the policy solver (Algorithm 1) within the learning process, so that the policy must reason about the same constraints during training and testing. This ensures the feature distributions at training and testing time are similar: the topological graphs (and thus the action set) are similar, the available landmarks obey the same constraints, and more generally the state distributions induced during training and testing time are close to each other.

Furthermore, as we will show next, training the policy in unknown environments means it will make mistakes and end up in states not visited by the expert, and we will be able to provide demonstrations on how to recover from these errors.

4.3 Learning to Recover from Mistakes

In a traditional supervised imitation learning setting, the learning algorithm would optimize the loss in Equation (4.5) over all states provided in the demonstration (i.e. the states visited by the expert). However, Ross and Bagnell [126] showed that this may lead to poor performance at test time due to a mismatch between the distribution of states encountered by the policy at training time and the distribution of states induced by the policy at test time. Because the learned policy's predictions affect future states and observations, states encountered at test time may be vastly different from states encountered during training. To give a concrete example in our setting, a small mistake (going through an incorrect door) leads to a completely different distribution of states than the demonstrations (a different room/hallway with different landmarks). Since the policy has not seen states like these during training, it cannot decide how to recover. This causes small errors to compound, and even a small mistake (the wrong turn) can result in arbitrarily bad performance in our setting (being in the wrong part of the map). In essence, since the demonstrations only contain successful examples, the learning process never receives any demonstrations of errors that are likely to happen while the robot is following directions.

To learn to recover from mistakes, we must include demonstrations of making errors and how to recover. Since the demonstrations do not include examples of making mistakes, we collect training data from traces of the current learned policy (instead of traces of the expert's policy). We then minimize the loss in Equation (4.5) over that training data, and iterate. This powerful idea was first introduced by Ross et al. [129] in a meta-algorithm called DAgger. The intuition is that the learned policy will make mistakes that we can observe (the error), and the expert's policy will provide the correct demonstrated actions (the error recovery). As the learning algorithm iterates, it will build up demonstration of all inputs the policy is likely to observe during execution, as shown in Algorithm 2.

This approach learns a policy that does well on the distribution of states induced

Algorithm 2 Dataset Aggregation for learning a policy to follow directions over N iterations. Each iteration applies the current policy to the all training directions Z_{train} using FOLLOW_DIR (Algorithm 1).

```
procedure TRAIN_POLICY($Z_{\text{train}}, N$)
    Initialize $\pi$ to any policy in $\Pi$ parameterized by $w$
    for t = 1 to N do
        Generate a trajectory of state-action pairs $D = \{s, \pi(s)\}$ by applying
        the current policy to all directions in the training set.
        Compute the expert's actions ($a^* = \pi^*(s)$) for all states in $D$.
        Minimize $\ell(s, a^*, w)$ over all accumulated examples by computing $\partial\ell/\partial w$.
        Update policy according to Equation (4.8).
    end for
    return $\pi$
end procedure
```

by the learned policy, instead of only the distribution of states visited by the expert. In the context of following natural language directions, we use the current policy to collect a set of training state-action pairs (by following each direction in the training set), and then compute what the expert's demonstrated actions a^* would have been for those visited states. We compute the loss for those examples, update the policy according to Equation (4.8), and continue iterating until the policy converges. We illustrate this iterative learning process in Figure 4.1. At each iteration the policy improves its ability to follow directions by increasing the cost of incorrect actions and decreasing the cost of the demonstrated actions. Since the state-action trajectory D is generated by the current policy, it may include states the expert did not visit. To compute the demonstrated action for *any* state, we will need a way to compute the expert's policy from the demonstration paths.

4.4 Computing the Expert's Policy

To learn a policy that can recover from mistakes, DAGGER requires the ability to evaluate $\pi(s)$ and $\pi^*(s)$ for any state s. The current policy is easy to evaluate for any possible state, by simply enumerating the possible actions from that state and computing costs (see Equation (3.9)). However, since the demonstration data consists of *paths*, we only have demonstrated actions for states along each path. Learning

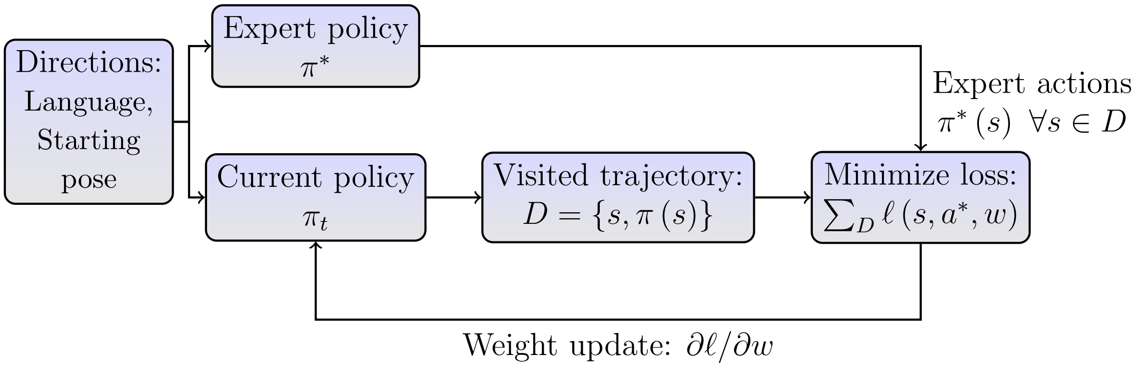
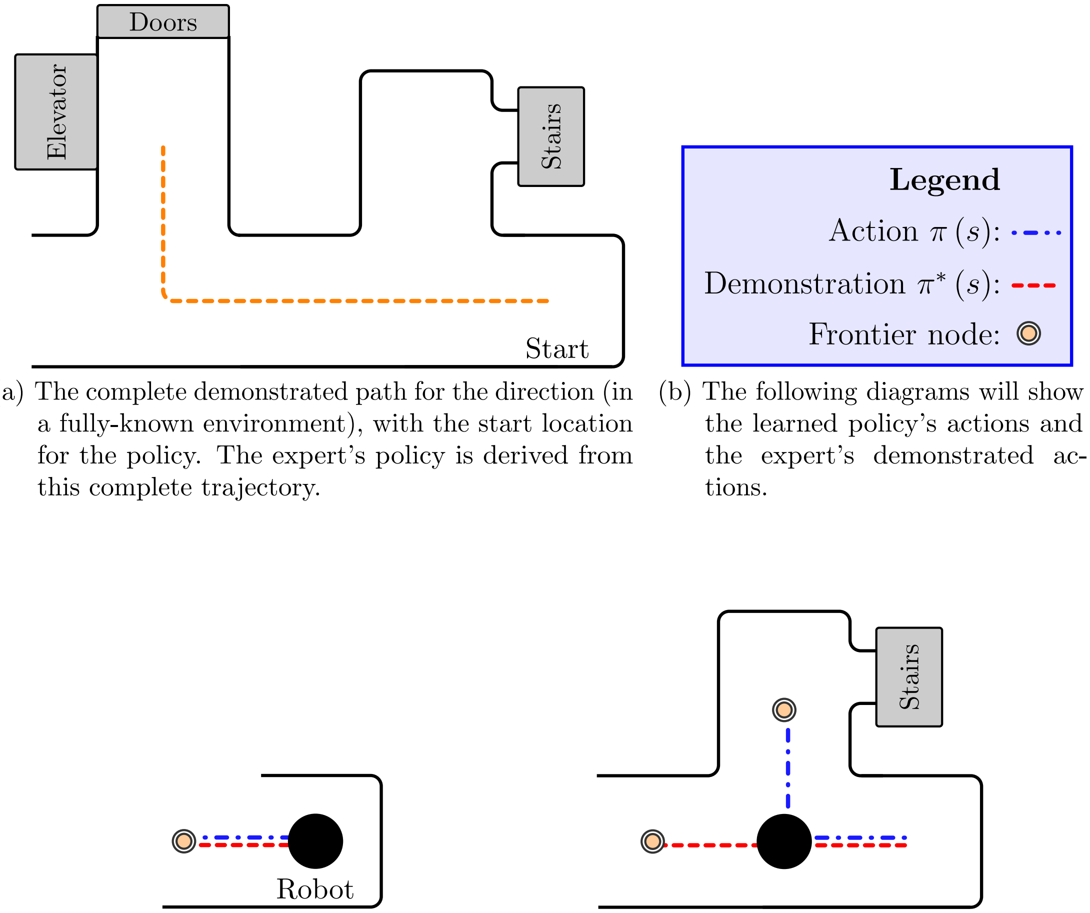


Figure 4.1: Iterative imitation learning formulation: To learn the policy π, we follow a direction to generate trajectories of visited state-action pairs. We then use the expert policy π^* to compute the demonstrated actions for the visited states, minimize the loss function that penalizes disagreements, and update the policy weights.

the policy using Algorithm 2 requires the ability to evaluate $\pi^*(s)$ for any state, not just the states along the demonstration path. While we do not need access to the expert's cost function, we do need knowledge of which action the expert would select from any given state (i.e., the choice the expert's policy would make). In principle, this means that training a policy with DAgger would require access to the expert during the entire procedure, to query what the action should be if the policy visits a state not along the demonstration path. Since this would be very time-consuming, we instead make the assumption that a person following directions would realize when they become lost, backtrack, and continue with the correct action sequence. By keeping track of the point of divergence between our policy and the expert's path, we can determine which actions do not match the expert's and what the expert's action should be. We illustrate this process in Figure 4.2, where our approach uses the complete expert's path to extract the expert's decisions.

This enables the learning algorithm to extract the expert's action from any given state visited by the current policy, even if the expert did not visit that state. This approach allows us to compute the expert's demonstrated action for any state visited by the policy, even if that state was not visited by the expert. Because we can compute the correct action the policy should have taken if it makes a mistake at training time, the policy can then learn how to recover from mistakes by including training examples that show failures (and how to recover from them). For example, in



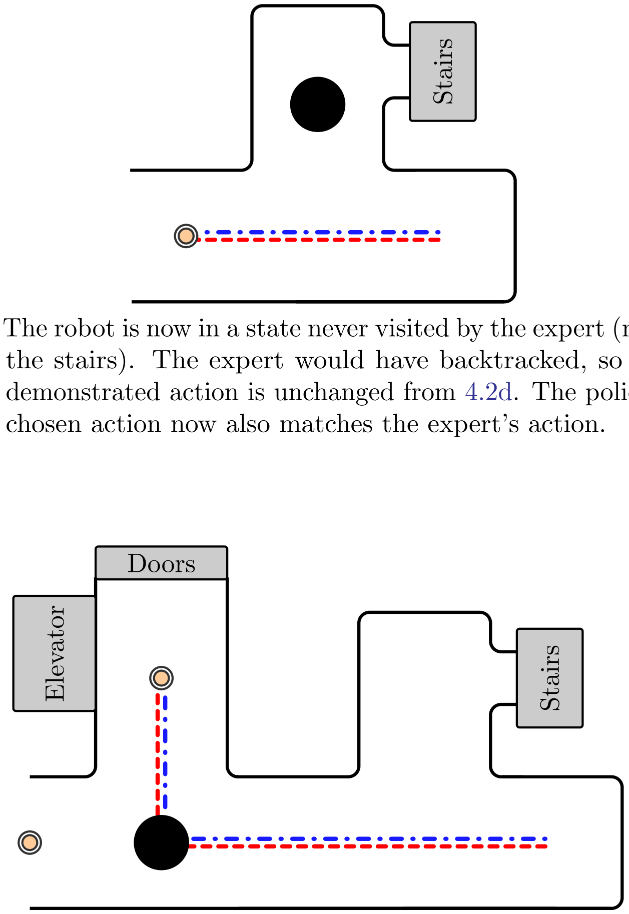
(a) The complete demonstrated path for the direction (in a fully-known environment), with the start location for the policy. The expert's policy is derived from this complete trajectory.

(b) The following diagrams will show the learned policy's actions and the expert's demonstrated actions.

(c) First action: the computed and demonstrated actions match.

(d) Second action: the policy turns right too early, whereas the demonstrated action (from the expert) is to keep going straight.

Figure 4.2: Figure continues on next page.



(e) The robot is now in a state never visited by the expert (near the stairs). The expert would have backtracked, so the demonstrated action is unchanged from 4.2d. The policy's chosen action now also matches the expert's action.

(f) After the robot backtracks, both the robot and expert policies choose to turn right.

Figure 4.2: Sequence of computed (-·-·-) and demonstrated (- - -) actions, to illustrate how we generate training examples for DAGGER. The full expert path turns right at the elevator, shown in 4.2a. In most cases computing the expert's policy is straightforward given the complete path (figs. 4.2c and 4.2d). When the policy does visit a state not seen by the expert, we can compute the expert's demonstrated action by keeping track of the divergence point (e.g., going straight in 4.2e). This approach provides more training data (by including failures) and enables the policy to learn how to recover from errors.

Figure 4.2, we store all the decisions as training examples, including the cases where the policy made a mistake by turning right too early (along with the demonstration of the correct action from that state). If we had trained our policy using only the states visited by the expert (in the traditional supervised imitation learning setting), we would have no examples of what to do when the policy takes us to a place not seen before, and much less training data overall. We now present results showing that imitation learning can be used to learn policies that can recover from errors.

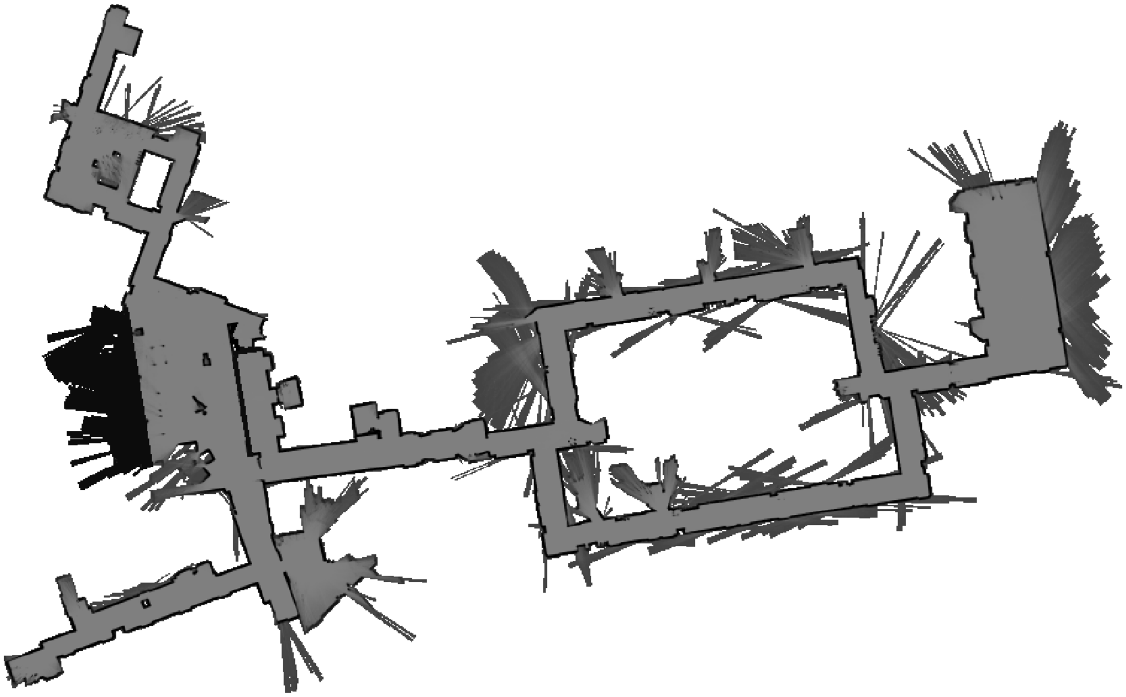


Figure 4.3: Map of Stata Center at MIT.

4.5 Results on a Corpus of Indoor Directions

We now apply our approach on natural language directions through one floor of an indoor building on MIT's campus (shown in Figure 4.3). Before describing our results of evaluating a learned policy through this unknown environment, we will first provide some details on the environment and corpus of directions used for these results. We will then present quantitative and qualitative results of applying our approach to the problem of following natural language directions through an unknown environment.

4.5.1 Methods

To better evaluate the performance of our approach of using imitation learning to learn to follow directions, in this section of results we abstract away the online map building and perception and focus on evaluating the learned policies in isolation. We will show end-to-end results on robot platforms that integrate online mapping and perception in Chapters 7 and 8. For these results we use a full semantic map that we incrementally "reveal" to a simulated robot as it travels in the environment. Although this model is a simplification of reality, it is still realistic under our scope assumption made in Chapter 1: the simulated sensor must still obey line of sight and sensing range constraints (computed using the metric map), and the simulated robot still uses a partial topological graph consisting of visited and frontier nodes. For this environment, we use a map of MIT's Stata center collected by Kollar [72]. This map consists of an occupancy grid, a topological graph of the corridors and rooms, and 211 semantically annotated objects (labels and geometry). Our simulator incrementally builds up a partial map that becomes available to the policy as the simulated robot travels through the environment. Another benefit of training in a simulated environment (while observing all physical constraints) is that we can train and evaluate the policy quickly; running multiple iterations of our imitation learning approach on a real robot would be time consuming.

The corpus of directions used for the results in this section are 40 multi-SDC directions that travel large distances through the map. We generated this corpus with the goal of remaining close to the corpus of directions collected by Kollar [72], while being able to focus on simpler directions free from hierarchical clauses or other complex linguistic structure. See Appendix A.1 for more details on the individual directions and aggregate statistics for the entire corpus. Each direction in the corpus consists of multiple SDCs, and include multiple verbs ("turn right," "turn left," "go," etc.), spatial relations ("towards," "past," "through"), and refer to a variety of landmarks. Some of the individual steps do not refer to a landmark (e.g., "Turn right."). In many cases, the landmark referenced in the direction may not have the same name as the object in the environment: for example a "sofa" or "seat" in the directions may refer to a "couch" in the environment. While these directions are somewhat simplistic, a person following these directions in an environment they have never been in before would almost certainly reach the correct destination. We will discuss possibilities for dealing with complicated directions in our future work (Chapter 9).

We will show results of our approach by training the policy on a subset of the directions, and testing on the rest. We split the data into a training and testing set using two regimes: fixed and random. In the fixed regime, we split the corpus of directions based on where each direction begins in the map: the training set consists of all directions starting to the left of a given coordinate, and the testing set consists of the rest of the directions (those starting to the right of that coordinate). We selected the split point such that we have 20 training and 20 testing directions. In the random regime, we randomly sample (without replacement) 20 training directions and 20 test directions.

Training

At each iteration of the learning process (see Algorithm 2), we apply the current policy to all directions in the training set. Each direction consists of a natural language command parsed into a sequence of Spatial Description Clauses (SDCs) and the complete ground truth path for the direction. Beginning at the starting pose in the path, the policy then follows the direction using Algorithm 1, taking a sequence of actions it believes best agree with the natural language command (under the current cost function). This results in a sequence of state-action pairs visited by the current iteration of the policy (not necessarily only states visited by the expert). As described in Section 4.4 and Figure 4.1, the algorithm then computes the expert's action for each state visited by the policy, then updates the weight vector by computing the subgradient of the loss function (effectively penalizing disagreements between the expert policy and the robot policy at the current iteration). This provides the necessary policy update, and we iterate this process.

During training we solve individual SDCs independently (beginning from their respective starting poses) instead of the entire sequence of SDCs in the direction. This is primarily for efficiency: it ensures we receive sufficient training data for all SDCs in the direction (especially the later SDCs). Training on complete directions would mean the policy must learn to follow each SDC correctly before it can get training data for the next SDC: a mistake early in the direction would result in no training data for SDCs later in the direction. This would require more learning iterations for the policy to converge on the entire training data. We have furthermore parallelized the training process so that the learning algorithm can solve each direction on a separate processor. The total training time for all training directions is a few minutes.

A benefit of our approach is the relatively small number of tuning parameters required, all of which are in the learning process. For the results that follow, we used $N = 16$ iterations of DAgger, and a learning rate for the weight update rule Equation (4.8) that decreases with:

$$\alpha_t = t^{-\gamma}, \tag{4.9}$$

for $\gamma = 0.9$. We also used a regularization parameter $\lambda = 1 \times 10^{-4}$. We determined these values using cross validation, but did not find large differences across variations of these.

Testing

We evaluate the learned policy by using it to solve all directions in the testing set. Each direction again consists of a sequence of SDCs, but only the start pose for the robot is given (position and heading) instead of the full path. The policy once again follows each natural language direction using Algorithm 1, resulting in a sequence of state-action pairs for each test direction. In this case, we solve the entire sequence of SDCs starting from the beginning: when the policy selects the stop action it transitions to the next SDC in the sequence and begins solving it from the current pose. When the policy selects the stop action on the final SDC it has completed following the direction, and we take that final pose as the ending location for the entire direction. Note that both the training and testing phases are applying the policy in essentially the same fashion (other than loss-augmentation and access to the full ground truth path during training). Training and testing are very similar: the learning phase can be thought of as iteratively testing (i.e. solving) the policy, where the weights change at each iteration.

Our main performance metric is the **mean ending distance error** from the true destination (averaged across all held out directions). In other words, we compute the distance between where the policy ended the direction (declared stop on the final SDC) and where the ground truth path ended. A smaller average error means the policy followed the directions more accurately. Additionally, we will compute a **success rate** over all held out directions, using different ending distance threshold (for example, 5 m). This means a direction is considered successful if the ending distance is less than the threshold. This additional success metric enables us to measure both aggregate results while being robust to outliers. For example, one direction could have a large error and increase the average ending error significantly, but the success rate will only reflect that one direction was incorrect.

4.5.2 Quantitative Results

We now present quantitative results over all held out directions, beginning with overall results on the entire corpus in the fixed regime. We also perform a feature ablation on the same set of directions to identify the most important components in our approach. Lastly, we perform extensive cross-validation experiments in the random regime.

Overall Results

In the fixed regime (training and testing on a fixed set of directions), our learned policy is able to follow 85 % of the directions successfully, with an average ending error (over all directions) of 2.63 m. The results on the same dataset for other configurations, shown in Table 4.1, indicate that our complete trained policy performs significantly better than the supervised learning approach (training only on those states present in

Table 4.1: Validation results on held out set of directions.

Configuration	**Mean error (m)**	**Success rate (%)**[*]
Complete trained policy	2.63	85
Supervised learning[†]	7.12	75
Full semantic map	9.53	60
Random destination[‡]	32.43	6

[*] Percentage of directions that finish within 5 m of the destination.
[†] Training only on states visited by the expert demonstrations.
[‡] Expected results for a random vertex (assuming complete graph knowledge).

the demonstrated paths), both in terms of mean ending error and success rate. This is because our policy has learned to recover from errors (we will show one instance of this behavior in Section 4.5.3).

Interestingly, giving the policy access to the complete map decreased the performance, compared to our approach. We believe this is because the policy still has to make a sequence of valid actions (i.e. take small steps towards the goal), but could use objects anywhere in the map (that would otherwise be invisible). In some sense, real-life visibility constraints help the policy make a better sequence of decisions by obscuring distant objects. It is likely that a planning-based approach (one that could optimize a complete path) would perform better than our policy-based approach when the complete map is available. No surprisingly, selecting a random destination in the graph has high error and low success rate (computed in expectation over the set of testing directions).

Ablating features

To evaluate the importance of the various categories of features described in Section 3.4, we compared the full feature set with various ablations. To do this, we turned off a single group of features and then re-trained the policy over the same fixed regime (using the same training and testing directions across trials). Note that only a single feature category is removed at once (all other features are intact) per trial. The results for these ablation experiments, shown in Table 4.2 and Figure 4.4, demonstrate that all components of the feature representation contribute significantly to our approach's performance as turning off any single one results in decreased performance.

The most dramatic performance drop occurs if we remove the geometric features. This is not surprising since to understand the command, the policy must reason about the shape of paths, the geometry of objects, and the relationship between the two. Removing semantic similarity (linguistic) features does decrease performance according to both metrics. This demonstrates the usefulness of being able to reason about synonyms (e.g., sofa and couch) or objects that co-occur (e.g., kitchen and microwave). Lastly, the performance without feature combinations of these geometric and linguistic features shows that it is important for the policy to reason about geometry and semantics *together*.

These results also indicate that computing stop action features is a very important part of our approach. Without these features, the policy cannot reason explicitly about the last action for each SDC, or compare the final action with a canonical representation of the expected path. Without these features, performance decreases significantly compared to the complete trained policy. This demonstrates that reasoning about (and learning) when to stop is challenging yet necessary, and that the stop action features are an important component that is integral to the success of our approach.

We also removed some of the words available to the policy to evaluate the importance of reasoning about different commands. Removing either verbs or spatial relationships decreased performance, indicating that reasoning about both the commanded action (e.g., "turn left") and the relationship to the landmark (e.g., "towards") is important. Removing all verbs and spatial relations also resulted in decreased performance.

Cross-Validation Results

In addition to these fixed-regime results (using the same test train and test directions across all experiments), we evaluated our approach in the random regime: sampling a new set of training and testing directions for each trial. By re-training the policy at each trial, we can get a more detailed representation of the performance of our approach over many randomly-sampled sets of directions. This provides a more

Table 4.2: Effect of feature ablations on same validation set.

Configuration	**Mean error (m)**	**Success rate (%)** *
Full policy	2.63	85
No semantic similarity features	7.49	60
No geometric features	20.21	10
No feature combinations	5.71	65
No stop action features	9.63	40
No verbs	6.96	65
No spatial relations	7.50	70
No verbs or spatial relations	9.21	60

* Percentage of directions that finish within 5 m of the destination.

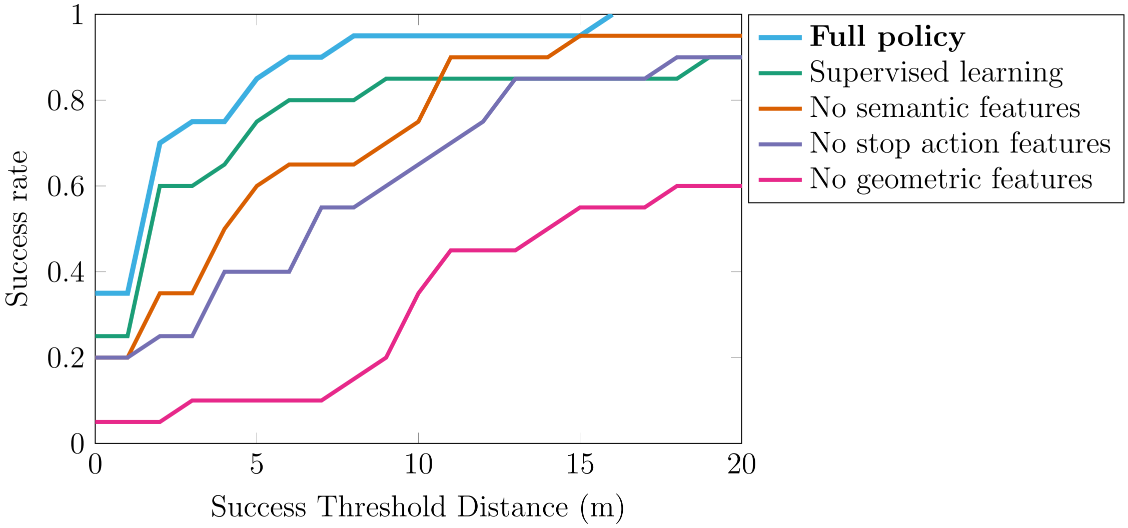


Figure 4.4: Success rate curves for different system ablations. The complete trained policy achieves the highest success rate for all possible distance thresholds.

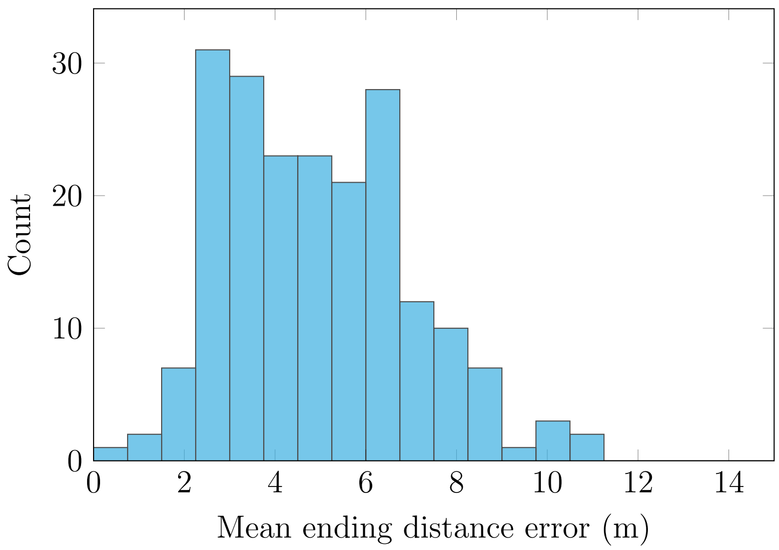


Figure 4.5: Histogram of mean ending distance error for the test directions, across 200 cross-validation trials. Lower error is better.

complete view of how well our approach deals with recovering from errors, handling various landmarks, and different parts of the environment.

The cross-validation results we present are over 200 trials of the random regime. We learn a policy on the training set, then evaluate it on the testing set according to the same metrics presented above. The histogram of mean ending distances (averaged per trial), shown in Figure 4.5, demonstrate that we are able to learn successful policies with a low average ending distance error (the distance between the policy's final action and the ground truth destination). Similarly, the average success rate (again, averaged across all test directions in each trial) in Figure 4.6 show that the learned policy is successful on a majority of the directions for most success thresholds. At a 5 m threshold, the trained policy averaged 78 % across all test directions in all trials, and this increases to 85 % at a 10 m threshold.

These results indicate that we are able to successfully learn a policy that follows natural language directions through unknown environments using only a relatively small number of directions (approximately half of the directions in the corpus). The learned policy is able to generalize to new directions, can reason about landmarks it has never observed, and is able to recover from mistakes. We next present qualitative results of applying our approach to directions through unknown environments.

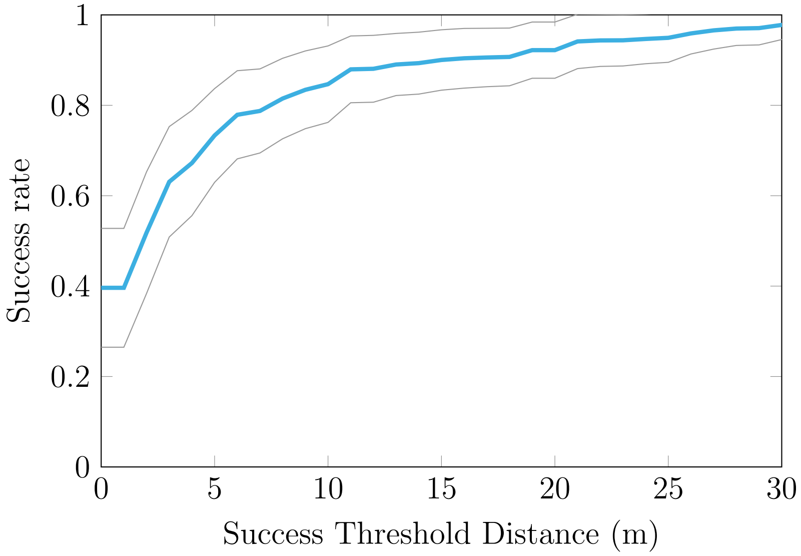


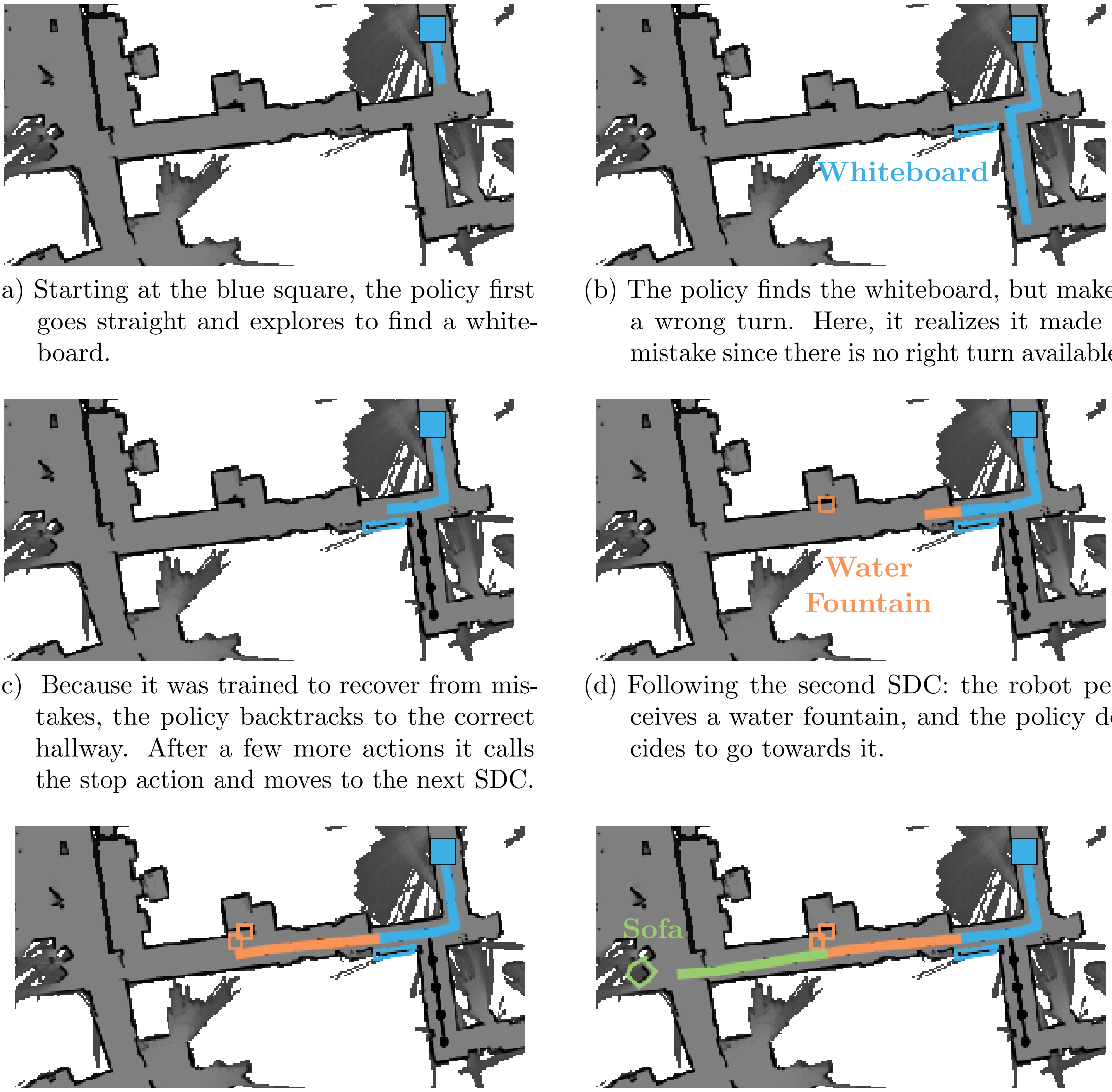
Figure 4.6: Success rate for all distance thresholds, averaged over 100 cross-validation trials. This plot shows the mean and standard deviation for the success rate. Higher success rate is better.

4.5.3 Qualitative Results

In addition to the aggregate results presented above, it is useful to consider individual directions and inspect (qualitatively) the policy, both during and after training. We now present a detailed sequence of actions for a single validation direction, and show the evolution of the policy during learning over two directions in the training set.

Detailed validation of a single direction

Qualitatively, the learned policy is able to make a complex sequence of decisions that follows the held out natural language direction shown in Figure 4.7. This direction has three SDCs and refers to various landmarks, one of which ("couch") is in the map with a different label ("sofa"). The learned policy begins by exploring the environment until it discovers the correct landmark (in other words the initial actions used no landmark). Although the policy makes a mistake, our approach to learning to recover from mistakes enables it to realize this when it reaches the end of the hall, and recover by backtracking to the correct corridor. Finally, the policy makes use of the stop action to declare when it is finished with the current clause and transitions to



(a) Starting at the blue square, the policy first goes straight and explores to find a whiteboard.

(b) The policy finds the whiteboard, but makes a wrong turn. Here, it realizes it made a mistake since there is no right turn available.

(c) Because it was trained to recover from mistakes, the policy backtracks to the correct hallway. After a few more actions it calls the stop action and moves to the next SDC.

(d) Following the second SDC: the robot perceives a water fountain, and the policy decides to go towards it.

(e) The policy perceives a second water fountain in the environment. After reaching this object, the policy considers the current SDC to be complete, calls the stop actions, and transitions to the third SDC.

(f) The policy associates a sofa in the environment with the "couch" in the direction. After reaching it the policy once again calls the stop action, and is now done following the directions since there are no more SDCs.

Figure 4.7: Sequence of decisions for following the direction "*turn right to the whiteboard, go past the water fountain, go to the couch.*" We color-code the actions and landmarks associated with each SDC, and show in black the previously visited nodes in the topological graph. Due to lack of space we only show selected decision points.

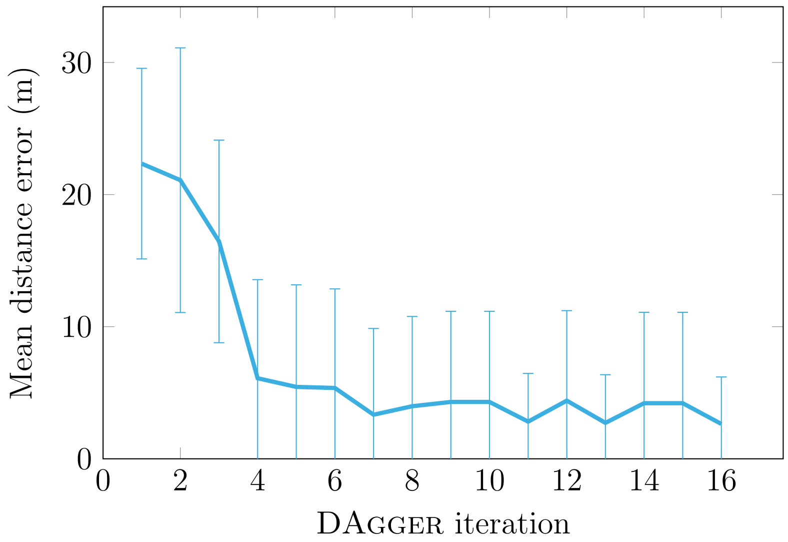


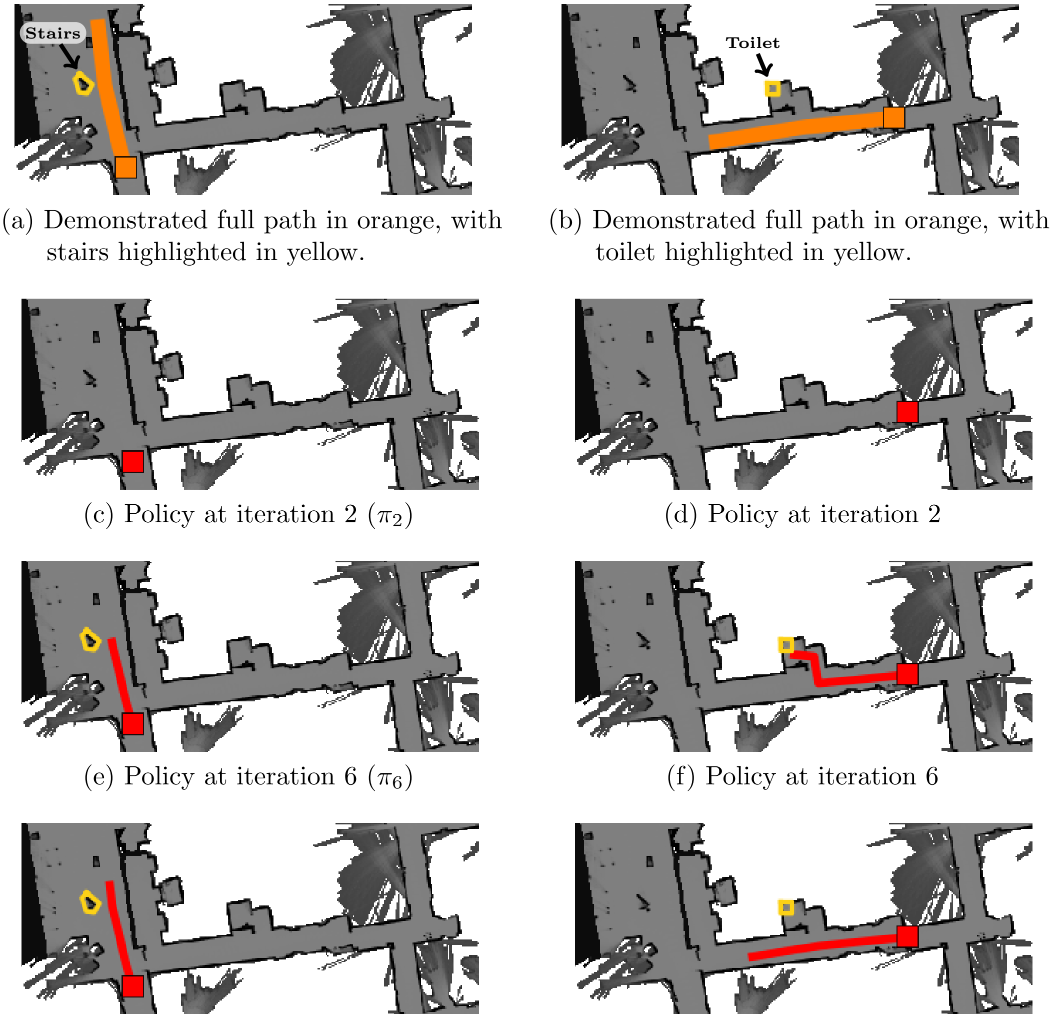
Figure 4.8: Distance loss (mean and standard deviation) for each iteration of DAGGER, computed over the same held out dataset.

the second SDC in the sequence. The policy continues with this SDC, then transitions to the final SDC. When the policy selects the stop action, it is finished following the direction. In this particular direction, the ending distance was 0 m for the final SDC, thus we consider the policy successful.

Evolution of the policy during learning

We first show the evolution of the policy *during* the learning process by measuring the error of the policy at that iteration, on the testing set of directions. In other words we are interested in the policy's best guess at each iteration of the learning process compared to the correct destination. In Figure 4.8 we see that the average ending error over the entire testing set is initially large, and decreases significantly in the first few iterations as the policy learns to understand the groundings for the different actions and landmarks in its training set. By iteration 10, the learning algorithm converges to a stable solution.

More qualitatively, we can observe the resulting paths of individual directions in the training set as it evolves over learning iterations. In Figure 4.9 we show the final action for the policy at multiple iterations, and see that the policy slowly learns the meaning of the word "past." At the second iteration the policy stops too early, but



(a) Demonstrated full path in orange, with stairs highlighted in yellow.

(b) Demonstrated full path in orange, with toilet highlighted in yellow.

(c) Policy at iteration 2 (π_2)

(d) Policy at iteration 2

(e) Policy at iteration 6 (π_6)

(f) Policy at iteration 6

(g) Policy at iteration 10 (π_{10})

(h) Policy at iteration 10

Figure 4.9: Evolution of policy (during training) over several iterations. The directions are "*Go past the stairs*" (left) and "*Go past the bathroom*" (right). We show the full demonstrated paths in 4.9a and 4.9b, starting at the square. The paths in 4.9c - 4.9h are the final actions resulting from their respective policies at the given iteration in the learning process, and show the object used by the policy (if any) in yellow. We can see how DAgger progressively learns the meaning of the spatial relationship "past" and landmark groundings, including associating the toilet present in the environment with the bathroom mentioned in the direction.

after more training examples it goes towards the landmark. The learning algorithm continues to provide corrective demonstrations, and the policy eventually successfully learns the meaning of the word “past” by the tenth iteration.

4.6 Chapter Summary

In this chapter, we discussed how to train the policy π by learning the cost function weights w from demonstrations of people giving and following directions. The demonstrations consist of natural language directions, and their associated paths and landmarks. From these data, we formulated an online learning problem that attempts to minimize the number of disagreements between the expert’s demonstration and the current policy, using a maximum margin approach. We used this formulation as the inner loop of the DAgger meta-algorithm, and at each iteration of the learning process compare all state-action pairs generated by the current policy’s decisions against those the expert would have taken. We showed how to obtain training examples of the expert’s demonstrated action for states *not* visited by the expert. This iterative learning process is simple, elegant, and most importantly fast. Using quantitative and qualitative results, we demonstrated that the learning method produces policies that can follow natural language directions through unknown environments and recover from mistakes.

Chapter 5

Inferring Maps and Behaviors from Natural Language

> What is best in music is not
> to be found in the notes.
>
> Gustav Mahler

Our approach so far has only considered the user's natural language command as a specification of a *task*, i.e., what the robot should do. In our problem of following directions, the user is commanding the robot to go somewhere in an unknown environment. Moreover, the natural language command may also provide information about the *world* in which the robot operates. For example, consider the command "go to the kitchen down the hallway" (Figure 5.1). If the robot can make use of the language constraints once it detects the hallway (namely, the fact that the kitchen is down the hallway), it can take more effective actions towards the kitchen (using the hallway) instead of relying on blind exploration to find the kitchen directly. When presented with these types of commands, a robot should reason about the approximate location of the landmarks in the world, using both information provided by its sensors and information contained in the language. As we will show, building a model of the environment from natural language descriptions will improve the robot's performance when following directions in unstructured unknown environments.

In this chapter, we propose a novel view of **language as a sensor**. We build

Figure 5.1: One possible natural language command for an autonomous wheelchair. Since the user may be able to see beyond the robot's perception range, they may be able to communicate information about the world *and* the desired task through a single natural language command.

upon the work of Chapter 3 where the robot builds a partial map of the world using its traditional sensors (such as cameras and LIDAR), and enable the robot to "fill in" the unknown parts beyond its sensor range with information implicit in the instruction. To accomplish this, we first extract annotations from the user's natural language command. These annotations are essentially facts about the world, inferred from language. We then learn a distribution over possible maps from these annotations, hypothesizing the locations of landmarks. For instance, in the example above, the robot would infer the location of the kitchen based on its knowledge of the hallway. The policy can then use this prior information to make more informed decisions about where to go next, given the current distribution of maps. As in prior chapters, the robot updates its internal representation of the world as it travels in the environment and receives new metric observations (such as the location of perceived landmarks). The robot continues to execute actions until the policy explicitly declares it has completed the command. By reasoning and planning in the space of beliefs over object locations and groundings the policy can reason about parts of the world that are not initially observed, which enables the robot to follow natural language directions through unknown environments with improved performance.

This chapter details our approach to inferring a distribution of maps that accounts for the robot's sensor measurements as well as the information implicitly contained in the language, and then inferring what the resulting constraints on the robot's behavior should be in this distribution. We begin with an overview of this approach in Section 5.1, where we factor the problem into three coupled problems: annotation inference, map inference, and behavior constraint inference. Section 5.2 then details the natural language understanding component that uses graphical models to convert free-form natural language into annotations (facts about the world) and behavior constraints (what the robot should do). The annotations are analogous to sensor measurements, except they are generated from the language command. We use these annotations (as well as any sensor observations) in a semantic mapping framework that we describe in Section 5.3 to infer a distribution of maps.

The work presented in this chapter is a collaboration with Sachithra Hemachandra, Thomas Howard, and Matthew Walter, and is drawn from two previous publications: Duvallet et al. [42] and Hemachandra et al. [57]. We present a summary of the entire approach as a service to the reader of this dissertation. In this chapter we will only discuss using language to generate a distribution of maps and constraints on the robot's behavior that are sent to the policy. We will detail in Chapter 6 how the policy makes use of this information to generate action sequences, and how to learn a belief space policy using imitation learning.

5.1 Overview

In this chapter we formulate the problem of following natural language directions as one of inferring the robot's future trajectory $x_{t+1:T}$ that is most likely for the given direction Λ:

$$\underset{x_{t+1:T} \in \Re^n}{\operatorname{argmax}} \; p\left(x_{t+1:T} | \Lambda, z^t, u^t\right), \tag{5.1}$$

where z^t and u^t are the respective histories of sensor observations and odometry data. When a full map of the environment is available, this problem is solved by conditioning the distribution over the known world model. Without any a priori knowledge of the environment, we treat this world model as a latent variable S_t, and then interpret the natural language command in terms of the latent world model, resulting in a distribution over behaviors β_t. We then solve the inference problem in

Equation (5.1) by marginalizing over the latent world model and behaviors:

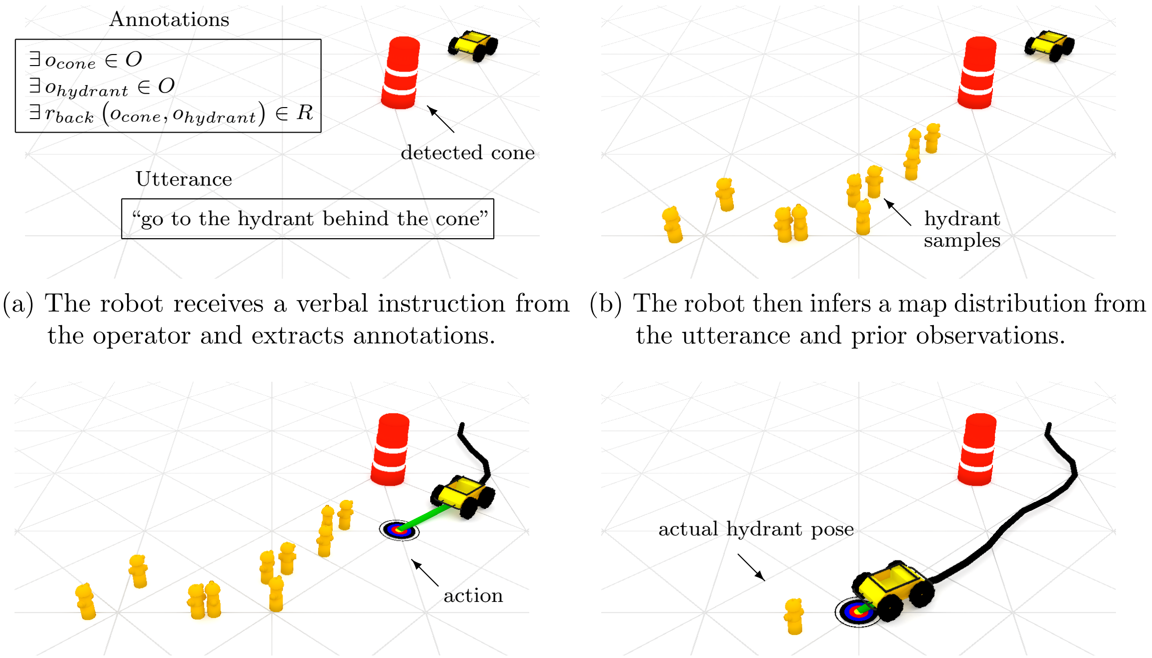
$$\underset{x_{t+1:T} \in \Re^n}{\operatorname{argmax}} \int_{\beta_t} \int_{S_t} p(x_{t+1:T} | \beta_t, S_t, \Lambda) \cdot p(\beta_t | S_t, \Lambda) \cdot p(S_t | \Lambda) \, dS_t \, d\beta_t, \tag{5.2}$$

where we now omit the measurement and odometry histories (z^t and u^t) for conciseness. Reasoning over the entire space of semantic maps and behaviors would be intractable, so we instead approximate Equation (5.2) by representing the latent map and behavior variables as discrete *samples* from the distribution in a particle filtering framework:

$$\underset{x_{t+1:T} \in \Re^n}{\operatorname{argmax}} \sum_{\beta_t^{(i)}} \sum_{S_t^{(i)}} p(x_{t+1:T} | \beta_t^{(i)}, S_t^{(i)}, \Lambda) \cdot p(\beta_t^{(i)} | S_t^{(i)}, \Lambda) \cdot p(S_t^{(i)} | \Lambda). \tag{5.3}$$

Each map sample $S_t^{(i)}$ represents one possible environment hypothesis, and each behavior sample $\beta_t^{(i)}$ is a set of action constraints within that map.

By structuring the problem in this way, we are able to treat following natural language directions as three coupled learning problems (visualized using the scenario in Figure 5.2). First, we infer a distribution over possible environments from the natural language command ($p(S_t|\Lambda)$ term above). As part of this, we extract annotations α_t from the command, representing our knowledge of the environment inferred from the language (Figure 5.2a). For instance, we may infer the existence of landmarks and their relative location to each other from the command. We treat these annotations as observations in a semantic mapping framework (along with data from the robot's sensors) to infer a distribution of possible maps (Figure 5.2b). Second, conditioned on the inferred world models we infer a distribution over behaviors ($p(\beta_t|S_t, \Lambda)$ term above). A behavior in this distribution is a set of action constraints, representing the *intent* of the command. Third, we use a learned belief space policy to sequentially solve for the actions that are consistent with the natural language command and the distributions of behaviors and maps ($p(x_{t+1:T}|\beta_t, S_t, \Lambda)$ term in Equation (5.2), Figure 5.2c). As in our prior approach, the robot executes one action and updates the world model distribution based on new commands or sensor observations. The policy continues to make a sequence of decisions (actions) until it explicitly declares that the robot is done following the direction (Figure 5.2d). Table 5.1 lists the additional symbols used in this chapter.


Annotations
$\exists\, o_{cone} \in O$
$\exists\, o_{hydrant} \in O$
$\exists\, r_{back}\left(o_{cone}, o_{hydrant}\right) \in R$
detected cone
Utterance
"go to the hydrant behind the cone"
hydrant samples
action
actual hydrant pose

(a) The robot receives a verbal instruction from the operator and extracts annotations.

(b) The robot then infers a map distribution from the utterance and prior observations.

(c) It takes an action (green) using the map and behavior distributions.

(d) This process repeats as the robot acquires new observations, refining its belief.

Figure 5.2: Visualization of one run for the command "go to the hydrant behind the cone," showing the evolution of our beliefs (the possible locations of the hydrant). The robot begins with the cone in its field of view, but does not know the hydrant's location.

Table 5.1: Symbol definitions for map and behavior inference.

α	annotation	set of objects, regions, and relationships
β	behavior	desired action and constraints
S	map	distribution of possible environment models

In order to better represent the space of possible commands (and the additional navigational information conveyed by the user), we introduce an extended version of the Spatial Description Clause from Section 3.1. This extended structure contains two additional fields:

- *Navigation Landmark:* a landmark name that (once seen) will provide navigation information.
- *Navigation Relation:* the geometric relationship between the goal and the navigation landmark.

These additional fields represent the information about the world contained in the command by constraining the location of the goal landmark with respect to a navigation landmark. For example, in Table 5.2 we show one sample command with implicit information (the kitchen is down the hall), and its decomposition into a sequence of Extended Spatial Description Clauses. The additional navigation information ("down the hall") provides more information about where the kitchen is, or can help resolve ambiguity if there are multiple matching landmarks. After an overview of the system architecture, Section 5.2 will focus on the natural language understanding module that converts the unstructured natural language direction to a set of annotations and behaviors.

System Architecture

The framework for the semantic map inference and learned belief space policy running on the robot is illustrated in Figure 5.3. As described above, the inputs are the natural language command and the robot's sensor observations. The Natural Language Understanding (NLU) module extracts annotations from the command: these represent the information about the environment conveyed in the language (for example, the existence of landmarks and their spatial relation to others in the world). The Semantic Mapping component then infers the possible world models efficiently by storing them as *samples* of maps from the distribution of possible environments. Each map in the distribution contains the location of landmarks, inferred from both sensor measurements (accurate) and the annotations (uncertain). Each map has an associated likelihood that changes over time, representing how well the map agrees with the robot's sensor observations as it moves and discovers new parts of the

Table 5.2: Extended Spatial Description Clauses for the command “Go past the kitchen that is down the hall and then turn right towards the doors.” Each semantically annotated clause contains additional information about the relationships between the goal landmark and other landmarks in the world.

	Λ_0	Λ_1
Verb	go	turn right
Landmark	kitchen	doors
Spatial Relation	past	towards
Navigation Landmark	hall	
Navigation Relation	down	

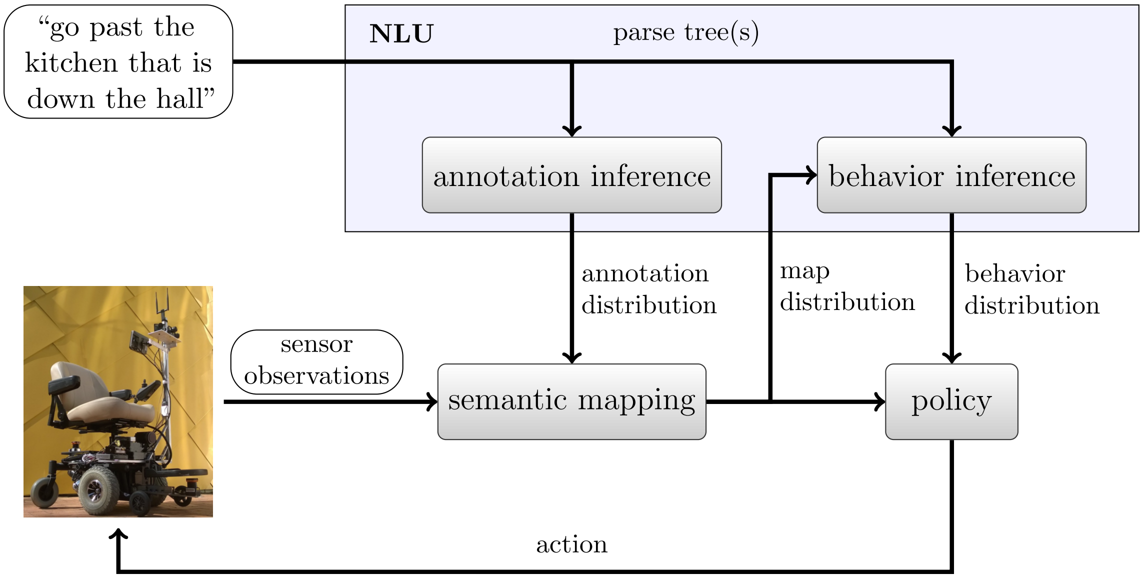


Figure 5.3: Outline of the semantic planning framework. The user provides a command to the Natural Language Understanding (NLU) module. First, this module extracts annotations, representing facts about the environment conveyed in the language. These are used by the semantic mapping module to generate a distribution of possible maps (including observations from the robot). The NLU module then interprets the command’s desired intent through a distribution of behaviors. The policy uses the map and behavior distribution to command a single action to the robot.

environment.

For each map sample, the NLU module then infers a behavior, representing the intent of the command with respect to the landmarks (known and hypothesized) present in the map. This effectively grounds the necessary landmarks described in the command. Lastly, the policy decides where to go, commanding a single action to the robot. During execution, the robot gains more information and each module reflects this new information: sensor observations update the map distribution, these updated maps change the behavior distribution, and the policy updates its desired next action using this information.

Currently, our implementation of the system only uses a single utterance from the user. However, our framework is general enough that we could convert further utterances into new annotations, and use those to update the map and behavior distribution. Implementing this ability to utilize language from the user *during* the robot execution remains future work.

5.2 Natural Language Understanding

Our framework learns models that identify the existence of annotations and behaviors conveyed by free-form language, and converts these into a form suitable for semantic mapping and the belief space policy. This is a challenge because of the diversity of natural language directions, annotations, and behaviors. We perform this translation using the Hierarchical Distributed Correspondence Graph (HDCG) model [59], which is an extension of the Distributed Correspondence Graph (DCG) [60] that offers improved efficiency. The DCG exploits the grammatical structure of language to formulate a probabilistic graphical model that expresses the correspondence $\phi \in \Phi$ between linguistic elements from the command and their corresponding constituents (*groundings*) $\gamma \in \Gamma$. The factors f in the DCG are represented by log-linear models with feature weights learned from a training corpus. The task of grounding a given expression then becomes a problem of inference on the DCG model.

The HDCG model employs DCG models in a hierarchical fashion, by inferring rules R to construct the space of groundings for lower levels in the hierarchy. At any one level, the algorithm constructs the space of groundings based upon a distribution

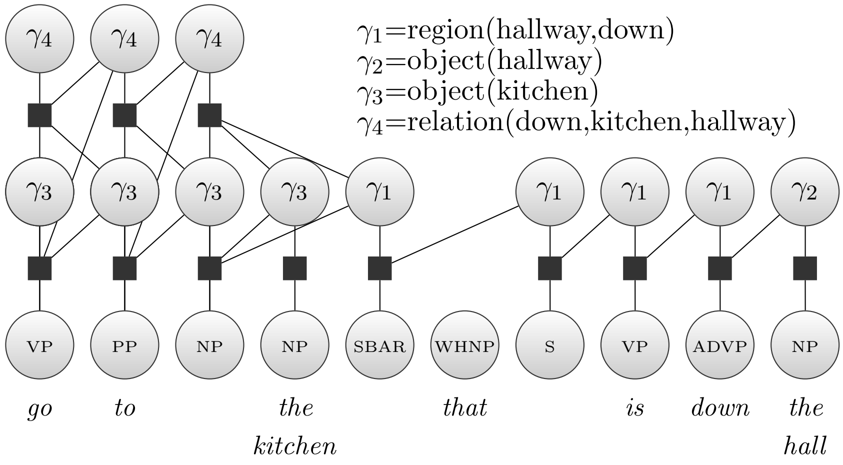


Figure 5.4: The active groundings in **annotation** inference for the direction "go to the kitchen that is down the hall". The two symbols at the root of the sentence (γ_3,γ_4) are sent to the semantic map to fuse with other observations.

over the rules from the previous level:

$$\Gamma \rightarrow \Gamma\left(\mathtt{R}\right). \tag{5.4}$$

The HDCG model treats these rules and the structure of the graph as latent variables. Language understanding then performs inference on the marginalized models:

$$\underset{\Phi}{\operatorname{argmax}} \int_{\mathtt{R}} p\left(\Phi | \mathtt{R}, \Gamma\left(\mathtt{R}\right), \Lambda, \Psi\right) \cdot p\left(\mathtt{R} | \Gamma\left(\mathtt{R}\right), \Lambda, \Psi\right) \tag{5.5}$$

$$\underset{\Phi}{\operatorname{argmax}} \int_{\mathtt{R}} \prod_{i} \prod_{j} f\left(\Phi_{i_j}, \Gamma_{i_j}\left(\mathtt{R}\right), \Lambda_i, \Psi, \mathtt{R}\right) \cdot \prod_{i} \prod_{j} f\left(\mathtt{R}, \Lambda_i, \Psi, \Gamma_{i_j}\left(\mathtt{R}\right)\right). \tag{5.6}$$

Annotation Inference

An annotation is a set of objects, regions, and relationships. We define a region as an object type paired with a spatial relation. A relationship is a spatial relation between ordered pairs of unique objects; the number of possible combinations of annotations is equal to the power set of the total number of symbols. The HDCG model infers a distribution of graphical models to efficiently generate annotations by assuming conditional independence of constituents and eliminating symbols learned

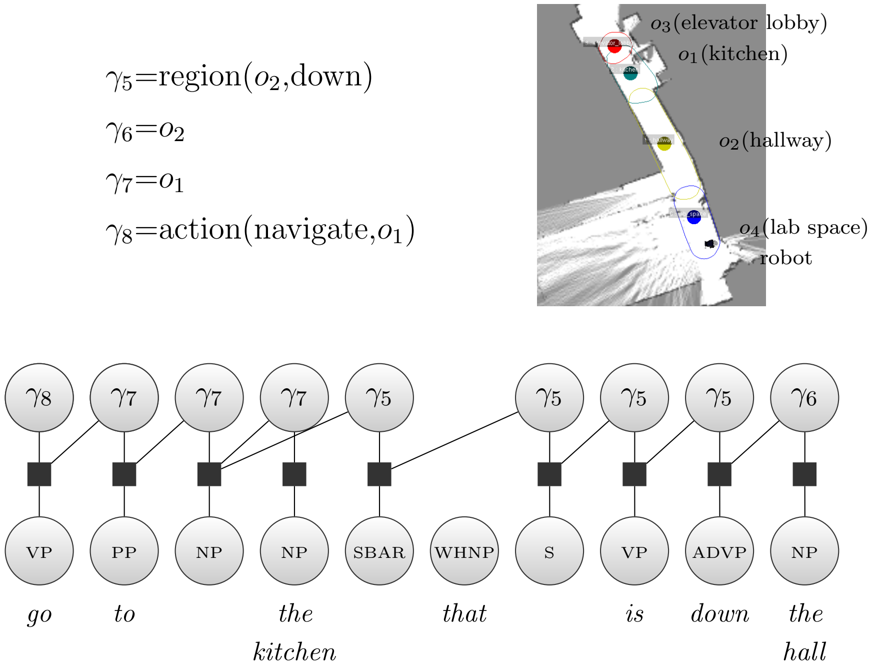
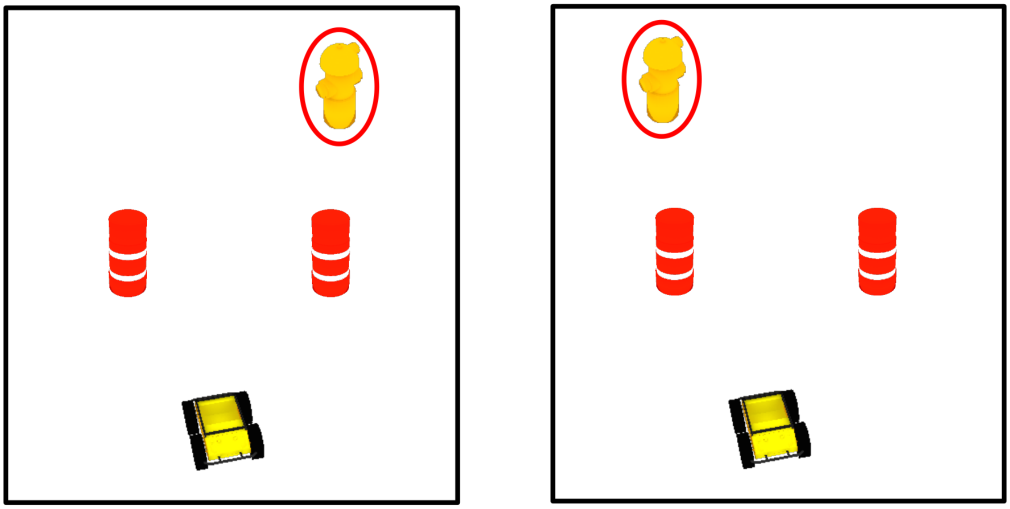

γ_5=region(o_2,down)
γ_6=o_2
γ_7=o_1
γ_8=action(navigate,o_1)
o_3(elevator lobby)
o_1(kitchen)
o_2(hallway)
o_4(lab space)
robot
γ_8 γ_7 γ_7 γ_7 γ_5 γ_5 γ_5 γ_5 γ_6
VP PP NP NP SBAR WHNP S VP ADVP NP
go
to
the kitchen
that
is
down
the hall

Figure 5.5: The active groundings in **behavior** inference for the direction "go to the kitchen that is down the hall" in the context of a inferred map with 4 objects. In this example a *navigate* action with a goal relative to o_1 would be sent to the policy planner.

to be irrelevant to the utterance. Figure 5.4 illustrates the model for the direction "go to the kitchen that is down the hall." In this example only 4 symbols (two objects, one region, and one relationship) are active in this model. At the root of the sentence are the symbols for a *down* spatial relation between a *kitchen* and *hallway* objects, and the *kitchen* object. The semantic map will fuse these annotations with other observations to infer a distribution over the possible environment models.

Behavior Inference

A behavior is a set of desired actions and path constraints that represent the *intent* of the natural language command. An action is represented using Extended Spatial Description Clauses (Table 5.2), specifying the objective of the natural language command. The constraints are spatial relations with respect to an object in the semantic map, and specify how the robot should perform the desired objective. Just



(a) Map with a hypothesized hydrant behind the left cone.

(b) Map with a hypothesized hydrant behind the right cone.

Figure 5.6: Two possible behavior groundings for the command “go to the hydrant behind the cone” in two different semantic maps, with the goal landmark circled in red. The cones in these maps are detected (known), while the hydrant is hypothesized. The behavior distribution enables us to reason about the desired behavior in the space of a semantic map distribution.

as in annotation inference, the space of groundings also grows as the power set of the total number of actions and constraints, so we again apply the Hierarchical Distributed Correspondence Graph to infer a distribution of behaviors. The HDCG model eliminates irrelevant action types, objectives, and spatial relations to efficiently infer behaviors. Figure 5.5 illustrates the model for the direction “go to the kitchen that is down the hall” in the context of an inferred map. In this example a *navigate* action with a goal relative to o_1 would be inferred as the most likely behavior for the policy planner.

As we maintain samples of the map distribution, we infer samples from the behavior distribution $\beta_t^{(i)}$ for each sampled semantic map $S_t^{(i)}$. The joint likelihood of each behavior and semantic map is:

$$p(\beta_t^{(i)}, S_t^{(i)}) = p(\beta_t^{(i)} | S_t^{(i)}) \cdot p(S_t^{(i)}). \tag{5.7}$$

By inferring behaviors for *each* sampled map, we explicitly interpret the meaning of the natural language command in the map distribution. For example, the two hypothesized hydrants in Figure 5.6 are the goal landmarks in their respective semantic maps. This combination of reasoning about possible maps and the resulting behavior distributions within these maps provides a probabilistic representation of likely maps and the command's intent within this map distribution.

5.3 Semantic Mapping

We represent the world model as a modified *semantic map* $S_t = \{G_t, X_t\}$ that consists of a metric and topological representation of the environment [168]. The topology G_t consists of a collection of nodes $\{n_i\}$ and edges denoting the connectivity between nodes. Nodes are spatial entities that represent either a region R_i or an object O_i, and can be observed by the robot or hypothesized using the language annotations. Each node has a label describing the type of region (e.g., kitchen, hallway) or object (e.g., cone, trash can). Regions are connected by edges when the robot traverses between two regions, or when language indicates the existence of a spatial relation between two regions (e.g., that the kitchen is down the hallway). Similarly, objects are connected by edges when the robot detects an object using its perception system, or when the command specifies a spatial relationship (e.g., "behind").

The metric map X_t is the vector of all poses x_i associated with each node n_i. The robot's perception system provides the location of objects and the spatial extent of regions as the robot drives in the environment. Regions represent spatially coherent areas in the environment that are intended to be compatible with a human's decomposition of space (e.g., rooms and hallways).

The natural language command provides annotations indicating the existence of different objects or regions in the environment, along with their location relative to other entities. We formulate a distribution of world models consistent with these annotations (α) by treating them as observations in a filtering framework, and combine these with observations from the robot's sensors (z) to maintain a distribution over the semantic map:

$$
\begin{aligned}
p(S_t|\Lambda, z^t, u^t) &\approx p(S_t|\alpha^t, z^t, u^t) && (5.8a)\\
&= p(G_t, X_t, |\alpha^t, z^t, u^t) && (5.8b)\\
&= p(X_t|G_t, \alpha^t, z^t, u^t) \cdot p(G_t|\alpha^t, z^t, u^t), . && (5.8c)
\end{aligned}
$$

The factorization within Equation (5.8c) models the relationship between the topology (graph G_t) and the induced metric map (X_t) that mimics a pose graph [65]. We maintain this factored distribution efficiently over time with a Rao-Blackwellized particle filter, where we use particles to maintain a sample-based approximation to the distribution over the topology, and a Gaussian distribution to model the distribution over metric poses [39]. Each semantic map particle:

$$
S_t^{(i)} = \{G_t^{(i)}, X_t^{(i)}, w_t^{(i)}\} \tag{5.9}
$$

consists of a sampled environment topology $G_t^{(i)}$, a Gaussian distribution over poses $X_t^{(i)}$, and a particle weight $w_t^{(i)}$ that represents the map's likelihood.

We update the map distribution (Equation (5.8a)) in three steps. First, when the robot receives new observations (from sensors) or annotations (from language) we sample modifications to the graph that update the topology for each particle. Second, we use these sampled graph topology modifications to perform a Bayesian update to the pose distribution. Third, we update the weight of each particle based on the likelihood of generating the given observations, and resample as needed to avoid particle depletion. We now outline this process in more detail.

During the proposal step, we first add an additional node and edge to each sample topology that model the robot's motion u_t, yielding a new topology $S_t^{(i)-}$. We then sample modifications to the graph $\Delta_t^{(i)} = \{\Delta_{\alpha_t}^{(i)}, \Delta_{z_t}^{(i)}\}$ based on the most recent annotations α_t and sensor observations z_t:

$$
p(S_t^{(i)}|S_{t-1}^{(i)}, \alpha_t, z_t, u_t) = p(\Delta_{\alpha_t}^{(i)}|S_t^{(i)-}, \alpha_t) \cdot p(\Delta_{z_t}^{(i)}|S_t^{(i)-}, z_t) \cdot p(S_t^{(i)-}|S_{t-1}^{(i)}, u_t), \tag{5.10}
$$

where $\Delta_{\alpha_t}^{(i)}$ are the modifications based on natural language observations and $\Delta_{z_t}^{(i)}$ are the modifications based on the robot's sensor observations. This updates the proposed graph topology $S_t^{(i)-}$ with the graph modifications $\Delta_t^{(i)}$ to yield the new semantic map $S_t^{(i)}$.

These modifications can add or delete regions, objects, or edges in the graph. We sample the graph modifications from two independent proposal distributions for annotations α_t and robot observations z_t. This is done by sampling a grounding for each observation, and deciding the required modification to the graph implied by the grounding. We now describe the graph modifications induced by natural language annotations and robot observations.

Graph modifications from natural language

When the algorithm receives a set of annotations $\alpha_t := \{\alpha_{t,j}\}$ inferred from the natural language command, it samples a graph modification $\Delta_{\alpha_t}^{(i)}$ for each particle:

$$p(\Delta_{\alpha_t}^{(i)}|S_t^{(i)-}, \alpha_t) = \prod_j p(\Delta_{\alpha_{t,j}}^{(i)}|S_t^{(i)-}, \alpha_{t,j}). \tag{5.11}$$

An annotation $\alpha_{t,j}$ can imply the presence of spatial entities (a region or an object) as well as a spatial relationship between two entities. For example, the command "go to the kitchen" implies the presence of a kitchen, and the command "go to the kitchen that is down the hallway" implies a spatial relationship between the kitchen and hallway (as well as their existence). Our semantic mapping algorithm treats these two types of annotations differently, described in more detail below. In both cases, the result is a modification to the topological graph that incorporates any new information contained in the annotation.

For annotations that describe the existence of spatial entities (regions or objects), the algorithm updates the topological graph with nodes and edges that reflect the information conveyed by the annotation in each map particle and the current semantic map distribution. The algorithm first chooses whether to add a *new* entity to the map or ground the annotation to an *existing* entity. This step utilizes a Dirichlet prior to account for the fact that the annotation may refer to new or existing entities in the map. Specifically, as the number of entities in the map with the same label increases, the likelihood of sampling a new one decreases. The algorithm adds a new entity to the map by adding a new node to the topological graph (a region or object depending on the annotation type) and then samples a new metric constraint for the associated edge. The sampled metric constraint accounts for the parts of the environment the robot has previously detected to prevent placing new objects or regions in previously visited areas. If the algorithm associates the annotation with an existing spatial entity in the map (e.g., a previously detected object), there is no modification to the graph.

For annotations that express a relationship between pairs of spatial entities, the algorithm samples an update to the graph topology such that the map particle agrees with the information in the annotation (in this case, both landmarks and the spatial relationship between them). For example, “the kitchen that is down the hallway” implies the “down” relationship between a “kitchen” and “hallway.” This step attempts to sample groundings for both spatial entities in the current map, but will add new spatial entities if necessary. For example, if there is only a “hallway” in the map, the algorithm will sample a new “kitchen” region. Any sampled regions will obey the given spatial relationship (according to features of the location of the regions and the location of the robot at the time of the utterance). Additionally, the robot keeps track of the area in the environment it has already observed, and will sample new hypothesized spatial entities in areas it has not already visited.

Graph modifications from robot observations

As the robot travels through the environment, it receives two different types of sensor observations that will also update the topological graph: region observations and object observations. Conceptually, the graph modifications from sensor observations are similar to those from language annotations, although each sensor observation provides more precise information about the spatial entity than language does (e.g., its metric pose, spatial extent, or connectivity). When the semantic mapping algorithm receives a set of observations $z_t := \{z_{t,j}\}$ from the robot’s perception system, it samples a graph modification $\Delta_{z_t}^{(i)}$ for each particle:

$$p(\Delta_{z_t}^{(i)} | S_t^{(i)-}, z_t) = \prod_j p(\Delta_{z_{t,j}}^{(i)} | S_t^{(i)-}, z_{t,j}). \tag{5.12}$$

The observations $z_{t,j}$ consist of either region observations (z^R) or object observations (z^O). Our semantic mapping algorithm treats these two types of observations differently, described in more detail below. In both cases, the result is a modification to the topological graph that incorporates any new information from the robot's sensor stream.

The first type of observations provided by the robot's sensors are region observations. Each region is a spatially coherent area in the environment (e.g., hallway, room, kitchen, etc.) and the robot's sensors measure the spatial extent and semantic label for the region it is in currently. As the robot moves through the environment, each region observation results in one of three possible modifications to the underlying topological graph:

- If the robot is still in the same region (as the previous observation), it updates the current region's spatial properties without modifying the graph.
- If the robot revisits a known region, it also updates that region's properties and associates its current location with that region.
- If the robot receives a region transition observation (e.g., the robot moved from one region to another), it creates a new region and updates its spatial properties.

If the robot creates a new spatial region, it will update hypothesized regions that match the new region's label (e.g., detecting a hallway that was previously hypothesized), or resample the location of hypothesized regions with a spatial relation to the new region (e.g., a hypothesized kitchen that is "down" from the newly-observed hallway). These region-based modifications to the topological graph incorporates new information from the robot's sensor streams to update the hypothesized semantic map.

The second type of observations provided by the robot's sensors are object observations. When the robot detects an object in the world (e.g., a hydrant or cone), the robot's sensors measure the object's relative pose and type. If the object is already known in the map, we add an edge between the robot's current node and the object's node, encoding the metric constraint provided by the observation. If there is no known object in the map but we have a hypothesis for the same object type, we attempt to ground the observation to a hypothesized object, and update its location appropriately. This step additionally ensures that we do not delete a hypothesized object that would be unlikely for this observation (i.e. a hypothesized object that is very far away). If the map has no previous knowledge about this new detected object (observed or hypothesized), we add a new node to the map representing the object's location and metric constraint. With this process, the semantic map algorithm maintains a probabilistic distribution over each object's location.

Re-weighting particles

After proposing modifications to each particle, we perform a Bayesian update to the map's Gaussian distribution of metric poses. We then re-weight each particle to take into account the likelihood of the semantic map (S) generating the received language annotations (α) and robot observations (z):

$$w_t^{(i)} = p(z_t, \alpha_t | S_{t-1}^{(i)})\, w_{t-1}^{(i)} = \; p(\alpha_t | S_{t-1}^{(i)}) \cdot p(z_t | S_{t-1}^{(i)}) \cdot w_{t-1}^{(i)}. \tag{5.13}$$

This accounts for the difference between the graph proposal distribution (i.e. map modifications) and the resulting distribution after an observation. Intuitively, proposed maps that agree with sensor observations (a hypothesized landmark is confirmed by the robot) will receive a high weight, whereas proposed maps that do not agree (an expected landmark was not detected) will receive a low weight. If the particle weights fall below a threshold, we resample particles to avoid particle depletion [39]. The factoring of Equation (5.13) enables us to compute the likelihood of both observation types separately: language observations (annotations) and robot sensor observations (regions or objects). These are described in more detail below.

To compute the likelihood of annotations, we use the natural language grounding likelihood under the map at the previous time step. A particle with an existing pair of regions (or objects) that agree with the language constraint will have a higher weight than one without. We evaluate this using the spatial relationship specified in the annotation.

To compute the likelihood of region observations, we give a low likelihood to new regions that overlap with a previously traversed region. We also consider the region type associated with the current region and calculate the likelihood of generating this observation given the unobserved regions in the particle. Only hypothesized regions that are very close to the robot can influence the region type observation.

To compute the likelihood of object observations, we use both positive and negative information. We calculate the likelihood of observing the object at the given location using the relative constraint of the object's current location in the map. We also calculate the likelihood of *not* observing the object using the sensor's field of view and the pose of the hypothesized object relative to the robot. This results in a lower particle weight for sampled maps with hypothesized objects in locations inconsistent with the detected parts of the world.

Semantic Map Inference Summary

The robot's representation of the world reflects all information available to the robot, both from its sensor observations and the natural language annotations. The algorithm infers a distribution of world models consistent with this information, stored as semantic map particles. Each particle contains the environment's graph topology, a distribution over metric poses, and a weight indicating the likelihood of that map particle. The algorithm updates this map distribution in three steps:

1. It samples modifications to the graph topology based on annotations (from the natural language direction) and observations (measurements from the robot's sensors).
2. It updates the metric pose distribution of nodes in the topological graph.
3. It re-weights each map particle to reflect the likelihood of the updated semantic map given the information available to the robot.

The result is a set of particles that efficiently represent the robot's knowledge of the world *within* and *beyond* its sensing range: the map contains both detected and hypothesized spatial entities.

5.4 Chapter Summary

This chapter introduced a novel view of language as a sensor that can be used to build uncertain maps. Similar to our approach in previous chapters, the robot builds a partial map of the environment using its physical sensors, such as cameras and LIDAR. Then it "fills in" the unknown parts of the map using information about the world that is contained in the natural language direction. In other words, using a single command, the user can communicate both a *task* and a description of the *environment*.

Our solution is to divide the problem of reasoning about the unknown parts of the environment into three coupled learning problems:

- understanding the natural language command to generate annotations (facts about the world) and behaviors (what the robot should do),
- inferring a distribution of maps that is consistent with observations from the robot's sensors as well as the language annotations, and
- learning a belief space policy that makes a sequence of decisions using the distribution of maps and behaviors.

This deconstruction provides an efficient solution to the entire problem of understanding natural language directions in unknown environments, and allows us to separate the uncertainty present in each component.

This chapter described a factored inference problem over latent world models and behaviors that represents the above decomposition. Using the Hierarchical Distributed Correspondence Graph (HDCG), a probabilistic graphical model that exploits the grammatical structure of language, we can infer a set of annotations present in the language. These annotations represent facts that we infer from the language, namely, the existence of landmarks and relationships between them (for example, the kitchen is down the hallway). Because annotation inference does not require sensor observations, the robot could hypothesize a map of the environment based entirely on language.

The natural language understanding module infers a distribution of desired *behaviors* for the command: action constraints that represent the intent of the command. To reason about the more complex space of possible directions (those that include some navigational information), we extended the Spatial Description Clause formulation to include two optional additional fields: a navigation landmark and a navigation relation. These fields are used to (a) clarify which landmark is correct in the case of ambiguity, and (b) provide a helpful hint as to the goal landmark location by utilizing relations between landmarks. We use this additional information when hypothesizing a distribution of valid semantic maps and following the direction.

To infer this map distribution, we extended an existing semantic mapping framework to treat the annotations as sensor observations in a probabilistic SLAM filter. This generates a distribution of possible environments that we represent efficiently using a Rao-Blackwellized particle filter. The map distribution takes into account information about landmarks the robot has *perceived*, as well as landmarks that were *described* by the user in the natural language command. As the robot travels through the environment, our approach updates the distribution of maps to reflect new information (e.g., sensor observations). In addition to providing an informative interface that can explicitly represent the robot's belief, this map distribution provides extra information for the policy to use when following directions.

The next chapter will focus on reasoning in this space of map distributions to plan a sequence of actions that follow the direction. This is significantly more challenging as the policy must reason about many possible map hypotheses when evaluating each action instead of a single partially-known world.

Chapter 6

Reasoning and Learning in Belief Space

> Logic will get you from A to Z;
> imagination will get you everywhere.
>
> Albert Einstein

Inferring a distribution of semantic maps from the natural language command as described in Chapter 5 provides several key benefits to our approach for following natural language directions through unstructured unknown environments. First, it allows us to use relative information between landmarks as additional information when following directions (for example, the relationship "behind the cone" given in a command). Second, it provides an explicit and intuitive representation of the uncertainty present in the environment in addition to the information available to the robot, represented as a distribution of maps which can be shown to the user.

However, the policy must now reason in the *belief space* of maps – instead of a single partially-known environment – to make a sequence of decisions and follow the direction. This map distribution represents possible environment models that are consistent with all the information available to the robot; most importantly the maps contain the location of hypothesized landmarks relevant to the direction. The policy should guide its decisions using the additional information from the map samples. This is challenging because each action the policy considers must now take into

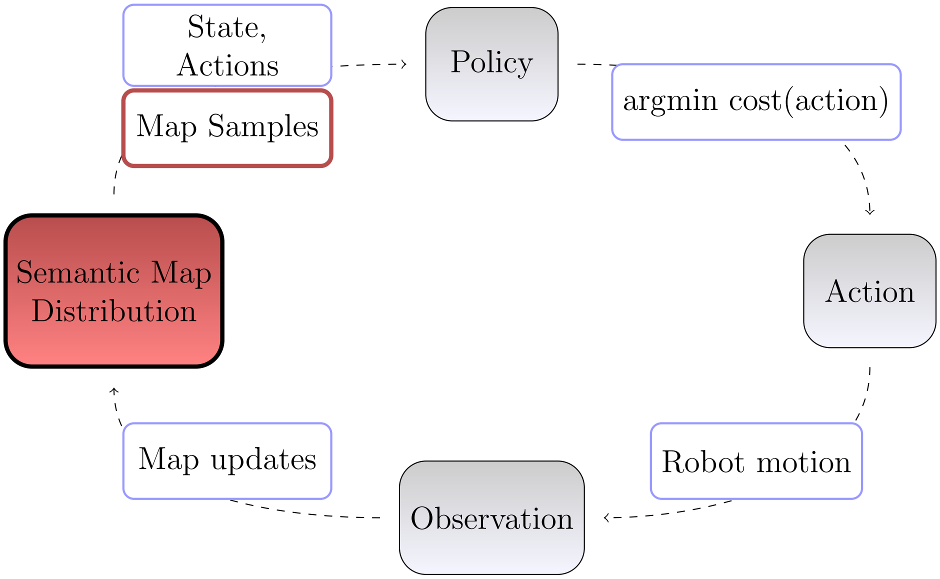


Figure 6.1: Belief space approach formulation as sequential decision making using a semantic map distribution: the robot infers a *distribution* of semantic maps, represented as map samples. The policy evaluates the cost of each action in the context of this distribution of maps, and chooses the one with the lowest cost. After the robot moves and receives a new observation, it updates its semantic map distribution. The policy repeats this process until it explicitly declares that it is done following the direction.

account a distribution of landmarks, which induces a distribution of features.

This chapter describes our belief space policy and how we train it. Our general approach for following natural language directions (as sequential decision making under uncertainty) remains largely unchanged: the policy selects a single action (out of many possible actions) to execute on the robot, and the policy continues making a sequence of decisions (updating its semantic map distribution) until it explicitly declares it is done. The main addition to the framework is that the robot infers a distribution of semantic maps using its sensor observations and the natural language commands, represented as samples in a particle filter (shown in Figure 6.1).

To utilize this additional information, we first present a belief space policy that reasons about the distribution of semantic maps to follow natural language directions in Section 6.1, by embedding the distribution of actions in a Reproducing Kernel Hilbert Space (RKHS). This enables us to project the distribution of features into a space where the policy can once again compute a cost function for each action. This approach is efficient and enables us to use a cost function to follow natural language directions in a distribution of possible environments.

We then describe in Section 6.2 a novel belief space imitation learning approach to train these belief space policies. Using the same demonstrated examples, we learn a belief space cost function that reasons explicitly about the distribution of possible environments. We present corpus-based experiments in Section 6.3 which show that our combined approach of inferring a map distribution and reasoning in belief space improves our performance on a corpus of complex directions. Furthermore, we will show that this approach can handle some types of ambiguous directions that our prior approach could not successfully follow.

6.1 Belief Space Reasoning

Our general approach to the problem of following natural language remains largely unchanged from that of Chapter 3. The robot builds up a mixed metric, topological, and semantic map of the world, except that it now predicts the parts of the world that lie beyond its sensor range (as described in Chapter 5). The policy will still enumerate the set of feasible actions using the topological graph (generating paths to visited and frontier nodes), and also considers the stop action (to represent finishing the direction). The belief space policy will evaluate the cost of each action, and execute the one with the lowest cost on the robot. Then, the policy will update its map distribution with any new sensor measurements or annotations. The robot will continue making decisions until the policy declares it has completed following the direction by selecting the stop action.

However, a key difference from the previous work is that the robot now maintains a *distribution* over possible semantic maps (instead of a single partially-known map), which induces a probability distribution of features per action. More specifically, whereas each action in Chapter 3 was a path paired with a *single* landmark, the belief space action in this chapter is now a path paired with a *distribution* of landmarks:

$$A_t = \{\text{path}\,(x, v) \in G_t \cup a_{\text{stop}}\} \times p\,(\mathcal{O}_t)\,. \tag{6.1}$$

Here the landmark distribution $p\left(\mathcal{O}_t\right)$ represents the semantic map's knowledge about where an object could be. For example, a landmark detected by the robot would have a very localized distribution of particles (low uncertainty), whereas a landmark hypothesized from language might have a larger spread of particles (high uncertainty). The intuition behind each belief action is the same: actions can explore, backtrack, or stop based on the path in the topological graph. However, each action is now reasoning about a distribution of semantic maps, representing the possible object locations for the landmark given in the command.

The belief space policy must make a sequence of decisions using the distribution of semantic maps $p\left(S_t\right)$:

$$\pi\left(x, p(S_t)\right) = \operatorname*{argmin}_{a \in A_t} c\left(x, a, p(S_t)\right). \tag{6.2}$$

This belief space action formulation means the policy must actually compare different probability distributions of features (one for each actions), and selects the one it believes will best follow the direction. This requires a way to represent and compute distances between probability distributions.

Fortunately, Hilbert Space embeddings have been shown to preserve the necessary information of these distributions while permitting efficient computation, which will enable us to learn from these distributions [105, 147]. Our policy will thus embed the action feature distribution in a Reproducing Kernel Hilbert Space using the mean feature map, which we now describe in more detail.

Belief Space Reasoning using Kernel Distribution Embedding

To reason about the *distribution* of landmarks when computing the cost of any action a in Equation (6.2), we embed the semantic map distribution in a Reproducing Kernel Hilbert Space (RKHS) [147, 148, 149]. The RKHS is a generalization of classical kernel methods to represent probability distributions (in our case, a distribution over action features induced by the semantic map distribution). For a good overview of Hilbert space embeddings for distributions, see Smola et al. [147].

We will thus represent our cost function as the mean embedding of a feature map [147]. More specifically, in our approach we first compute the features of an action in each map sample independently, then aggregate the resulting feature distribution

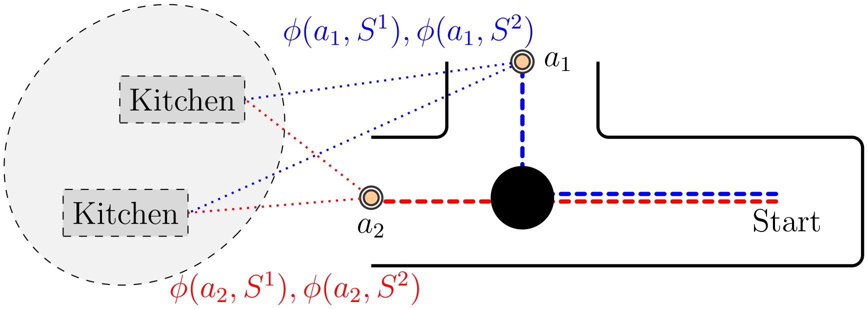

$\phi(a_1, S^1), \phi(a_1, S^2)$
a_1
Kitchen
Kitchen
a_2
Start
$\phi(a_2, S^1), \phi(a_2, S^2)$

Figure 6.2: Simplified illustration of computing feature moments in the space of hypothesized landmarks (in this case two kitchens). To compute the features over a landmark distribution, we compute the features of the action with each hypothesized landmarks. We then aggregate them by computing moment statistics across the distribution. For instance, to compute the cost of action a_1 we will aggregate the features across the two possible maps: $\phi(a_1, S^1)$ and $\phi(a_1, S^2)$.

by taking the first K moments of the features (across all map samples $S_t^{(i)}$):

$$\hat{\Phi}_1\left(x, a, S_t\right) = \sum_{S_t^{(i)}} p(S_t^{(i)})\,\phi\left(x, a, S_t^{(i)}\right) \tag{6.3}$$

$$\hat{\Phi}_2\left(x, a, S_t\right) = \sum_{S_t^{(i)}} p(S_t^{(i)})\,\left(\phi\left(x, a, S_t^{(i)}\right) - \hat{\Phi}_1\right)^2 \tag{6.4}$$

$$\dots$$

$$\hat{\Phi}_k\left(x, a, S_t\right) = \sum_{S_t^{(i)}} p(S_t^{(i)})\,\left(\phi\left(x, a, S_t^{(i)}\right) - \hat{\Phi}_1\right)^k \tag{6.5}$$

Intuitively, this computes features for the action and all hypothesized landmarks individually, aggregates these feature vectors, and then computes moments of the feature vector distribution (expected value, variance, and higher order statistics). We use the same features as those in Section 3.4 for individual landmarks. The process of computing aggregate features over a landmark distribution is illustrated in Figure 6.2, where the location of the kitchen is unknown but has two possible hypotheses (both used to compute features).

The cost function in Equation (6.2) can now be rewritten as a weighted sum of the first K moments of the feature distribution:

$$c\left(x, a, S_t\right) = \sum_{i=1}^{K} w_i^T \; \hat{\Phi}_i \left(x, a, S_t\right). \tag{6.6}$$

By concatenating the weights and moments into respective column vectors

$$W := [w_1; \ldots; w_k] \tag{6.7}$$

$$F := [\hat{\Phi}_1; \ldots; \hat{\Phi}_k] \tag{6.8}$$

we can rewrite the policy in Equation (6.2) as minimizing a weighted sum of the feature moments F_a for action a :

$$\pi\left(x, S_t\right) = \operatorname*{argmin}_{a \in A_t} W^T F_a. \tag{6.9}$$

Note that $W \in \mathbb{R}^{K \cdot d}$ where d is the feature vector dimensionality, so this increases the number of parameters to learn. As we will show next, we can apply a simple variant of the imitation learning approach from Chapter 4 to learn a policy that reasons in the map distribution.

6.2 Imitation Learning in Belief Space

We train the policy using imitation learning, by treating action prediction as a multi-class classification problem. Given an expert demonstration, we wish to correctly predict the expert's action out of all possible actions from the same state. Although Chapter 4 introduced imitation learning for training a policy to follow directions, it operated in partially-known environments without any prior over the undetected parts of the environment. In this chapter, we train the policy using a distribution of hypothesized maps to learn a belief space policy.

We assume the expert's policy π^* minimizes the unknown immediate cost:

$$C(x, a^*, S_t) \tag{6.10}$$

of performing the demonstrated action a^* from state x, under the current belief distribution S_t. However, since we cannot directly observe the true costs of the expert's policy, we must again minimize a surrogate loss that penalizes disagreements between the expert's action a^* and the policy's action a, using the multi-class hinge loss [34]:

$$\ell\left(x, a^*, c, S_t\right) = \max\left(0, 1 + c\left(x, a^*, S_t\right) - \min_{a \neq a^*}\left[c\left(x, a, S_t\right)\right]\right). \tag{6.11}$$

The minimum of this loss occurs when the cost of the expert's action is lower than the cost of all other actions, with a margin of one. This loss can be re-written and combined with Equation (6.9) to yield:

$$\ell\left(x, a^*, W, S_t\right) = W^T F_{a^*} - \min_a\left[W^T F_a - l_{xa}\right], \tag{6.12}$$

where $l_{xa} = 0$ if $a = a^*$ and 1 otherwise. This ensures that the expert's action is better than all other actions by a margin [121]. Adding a regularization term λ to Equation (6.12) yields our complete optimization loss:

$$\ell\left(x, a^*, W, S_t\right) = \frac{\lambda}{2}\|W\|^2 + W^T F_{a^*} - \min_a\left[W^T F_a - l_{xa}\right]. \tag{6.13}$$

Although this loss function is convex, it is not differentiable. However, we can optimize it efficiently by computing the subgradient of Equation (6.13):

$$\frac{\partial \ell}{\partial w} = \lambda W + F_{a^*} - F_{a'}, \tag{6.14}$$

for the best loss-augmented action a' at state s:

$$a' = \operatorname*{argmin}_a\left[W^T F_a - l_{xa}\right]. \tag{6.15}$$

Note that a' is simply the solution to our policy using a loss-augmented cost. This leads to the update rule for W:

$$W_{t+1} \leftarrow W_t - \alpha\,\frac{\partial \ell}{\partial w} \tag{6.16}$$

with a learning rate $\alpha \propto 1/t^\gamma$. Intuitively, if the current policy disagrees with the expert's demonstration, Equation (6.16) decreases the weight (and thus the cost) for the features of the demonstrated action F_{a^*}, and increases the weight for the features of the planned action $F_{a'}$. If the policy produces actions that agree with the expert's demonstration, the only weight update will be regularization.

Similarly to our approach from Chapter 4, we train the policy using the DAgger (Dataset Aggregation) algorithm [129], which learns a policy by iterating between collecting data (using the current policy) and applying expert corrections to the policy's decisions. Key to this approach is that we collect training information from all states visited by the policy, not just states that were in the demonstration. This enables us to learn a policy that does well on the distribution of states induced by the learned policy, instead of only the distribution of states that were visited by the expert.

Treating direction following in the space of possible semantic maps as a problem of sequential decision making under uncertainty provides an efficient approximate solution to the belief space planning problem. By using a kernel embedding of the distribution of features for a given action, we still reason about the distribution of landmarks in the semantic map. Using imitation learning for training the policy is simple, elegant, and requires no complex engineering of components or tuning of parameters.

6.3 Results

We evaluate our approach on a corpus of directions that expands upon the corpus presented in Chapter 4. This expanded corpus contains directions that include references to navigation landmarks. For example, the direction "go to the door across from the whiteboard" specifies *which* of many doors is the correct one. By comparing our semantic-mapping approach (with belief space reasoning) to our previous method (without any belief space reasoning) on this corpus, we will directly measure the benefit of reasoning in belief space.

We focus here on evaluating the performance of our approach to learning policies in belief space. To isolate this from the other components in the system, we again use a previously collected map of the environment, and "reveal" it to the robot as it travels in simulation (while obeying all line-of-sight and motion constraints).

Additionally, we approximate the semantic map sampling with a Gaussian distribution over landmark locations (with a mean and covariance per linguistic relation term, learned from data). We also approximate the belief likelihood update (re-weighting of particles) by invalidating hypothesized landmarks within the robot's line of sight. These approximations are once again an idealization of the entire system presented in Chapter 5 (especially with regard to perception), but in our experience the behavior is not significantly different from those on the integrated robot results we will present in Chapter 7.

Our methods here are almost identical to the results presented in Section 4.5. For the results that follow, we used $N = 15$ iterations of DAgger, and a learning rate for the weight update rule with $\gamma = 0.7$. The belief space policy uses $K = 2$ moments (the mean and variance) of the action feature distribution.

Comparison on extended corpus with complex directions

On the existing set of directions, the belief space approach improves the performance slightly. However, these directions were all relatively simple: they did not contain any additional navigation information. To evaluate the improvements of our belief space policy approach on directions that include the navigation landmark and relation fields introduced in Section 5.1, we generated an additional 15 natural language directions that contain information about the location of the goal landmark, especially in cases where this landmark could be ambiguous (e.g., there are many doors in the environment). These directions would (by design) be difficult to follow without using this additional navigation information.

For instance, the direction "go to the door after the water fountain, turn right, go straight to the cabinet," shown in Figure 6.3, contains information about the correct door to use (the one after the water fountain). Our semantic map inference approach can make use of this information when it generates a distribution of hypothesized maps: once the robot detects a water fountain the policy can generate possible door samples behind it. Utilizing this navigation information is especially important because there are many doors in the same hallway.

We now present experiments evaluating our approach to following directions (with belief space reasoning) on this extended corpus of direction. We begin with a

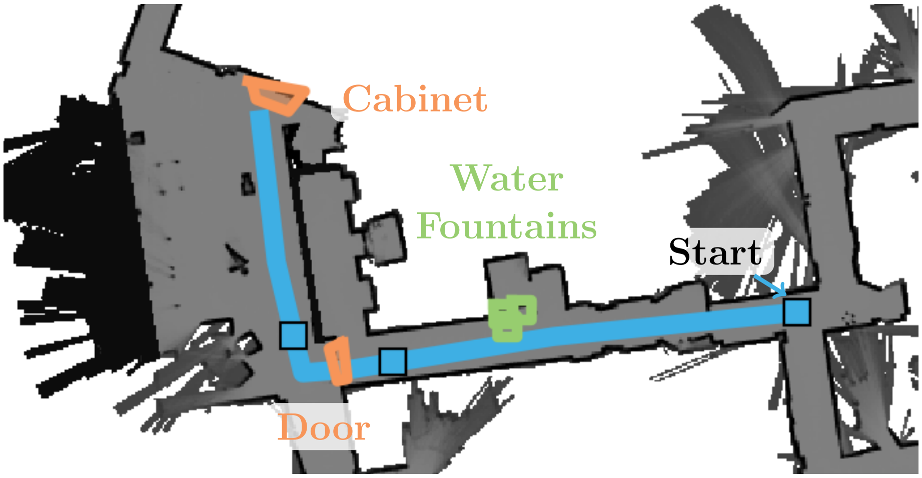


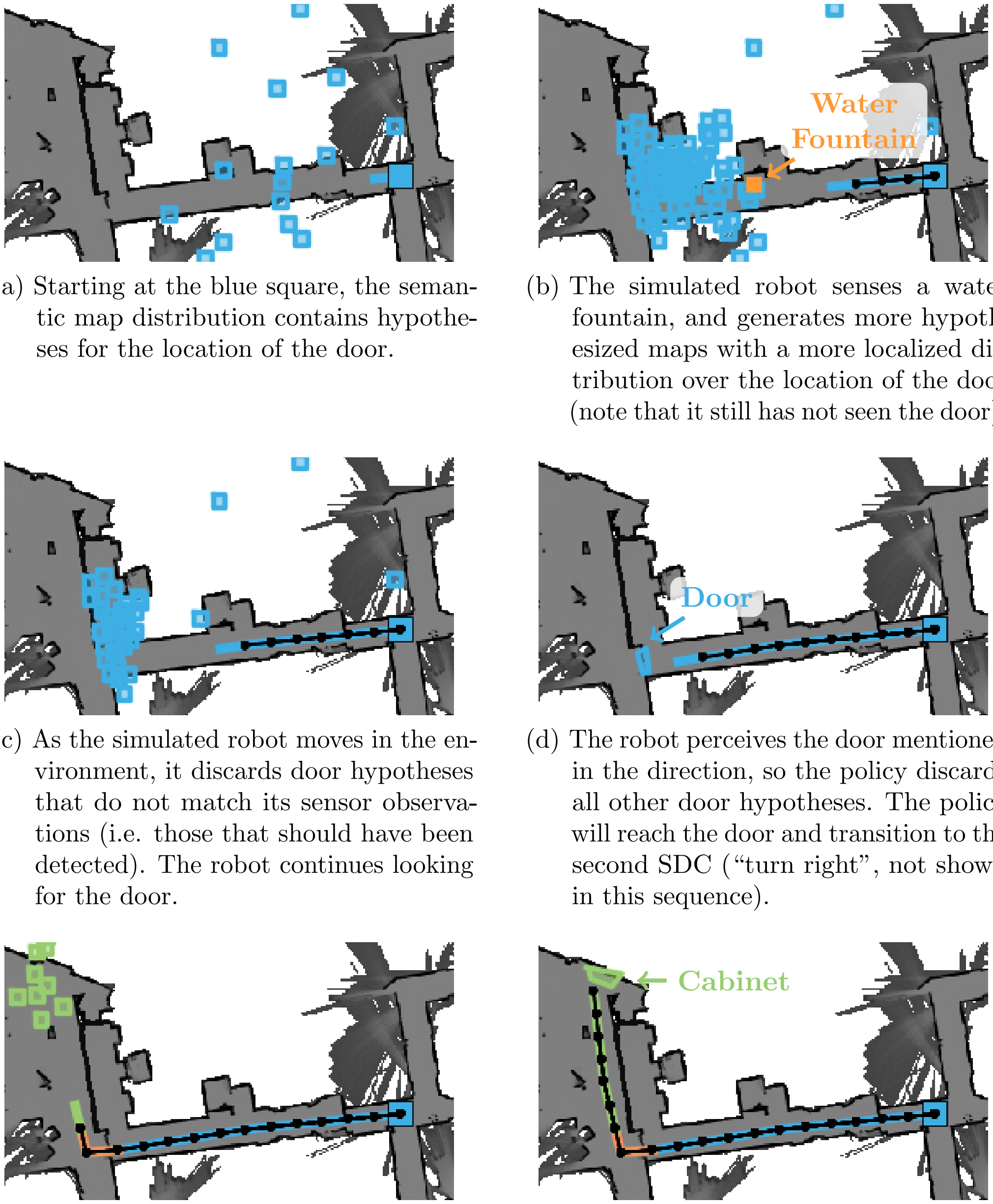
Figure 6.3: Ground truth path for the direction "go to the door after the water fountain, turn right, go straight to the cabinet." The direction contains the additional information about the location of the correct door (it is after the water fountain). Our belief space policy will leverage this additional information to guide its decisions, which is especially important because there are many doors in the same hallway.

qualitative detailed sequence of decisions for the example direction shown in Figure 6.3. The decisions of the policy in this case provide a good intuition for why this approach yields improved performance when following directions in unknown environments. We also compare this with the results (on the same direction) of our previous approach that does not generate possible maps, use a belief space policy, or use the navigation information.

We then look at quantitative aggregate results across many cross-validation trials, where we train the policy on some directions and apply it to the remaining held out directions. We will present results measuring both the average ending distance error and the success rate (for different distance thresholds). In both settings, the results of these experiments show that our belief space approach can successfully make use of the additional navigation information provided by the direction. This significantly reduces the average ending distance error, and improves the success rate.

Detailed validation sequence

We first look at the policy's sequence of decisions for one (held out) validation direction: "go to the door after the water fountain, turn right, go straight to the



(a) Starting at the blue square, the semantic map distribution contains hypotheses for the location of the door.

(b) The simulated robot senses a water fountain, and generates more hypothesized maps with a more localized distribution over the location of the door (note that it still has not seen the door).

(c) As the simulated robot moves in the environment, it discards door hypotheses that do not match its sensor observations (i.e. those that should have been detected). The robot continues looking for the door.

(d) The robot perceives the door mentioned in the direction, so the policy discards all other door hypotheses. The policy will reach the door and transition to the second SDC ("turn right", not shown in this sequence).

(e) The third SDC ("go straight to the cabinet") contains some information about the location of the cabinet (it is straight ahead). The policy generates a landmark distribution for its location.

(f) After the robot detects the cabinet, the policy continues going straight towards it, and here we show were it declared it was done following the direction (stop action).

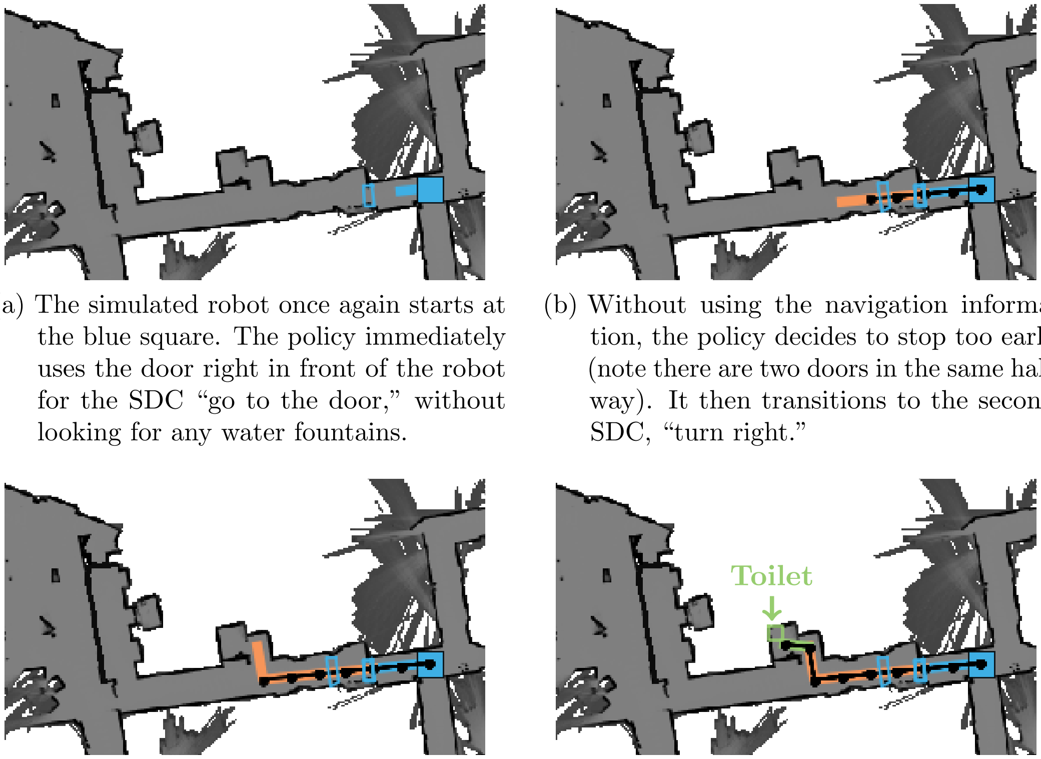
Figure 6.4: Sequence of policy decisions for the direction "go to the door after the water fountain, turn right, go straight to the cabinet." We show color-coded actions and landmark hypotheses per SDC, and show in black the previously visited nodes.

cabinet." The policy's semantic map distribution begins with very little knowledge, since the command only specifies the location of the door relative to a (still unknown) water fountain. This is reflected in the large distribution of hypothesized locations for the door (Figure 6.4a). The policy decides to go straight, exploring to find the door. As it moves, it also updates the semantic map distribution by invalidating samples that would have been visible from the robot's current location (since those hypotheses were incorrect). Once the robot detects a water fountain (Figure 6.4b), it generates more samples for the possible location of the door (using the relation "after"). This step highlights the use of the navigation information (namely, the door is after the water fountain) contained in the natural language instruction. By doing so, the policy now has much more information than it did previously to use when making decisions, and can look for the door in a more informed manner.

The policy continues going straight (ignoring a right turn into the bathroom), further updating the door landmark distribution (Figure 6.4c). When it finally detects the door (Figure 6.4d), it discards other hypotheses and finishes by going towards it. It then transitions to the second SDC (using the stop action), and begins executing the second SDC: "turn right". After two actions (not show in the figure), the policy again transitions to the next SDC: "go straight to the cabinet". Because the command does contain some implicit information about the location of the landmark (the cabinet is straight ahead), the semantic map distribution is more informative at the beginning (compare Figure 6.4e and Figure 6.4a). The learned belief space policy utilizes these samples to go straight until it sees the actual cabinet, then selects the stop action once it reaches the cabinet. Because there are no more SDCs, the policy has finished with the entire direction, with a 0 m ending distance.

The policy was successful in following this direction because it utilized all the information available in the language. First, it leveraged the direction's implicit information about the *world*, effectively using language as a sensor to build a map beyond the robot's sensor range. Second, the policy used the direction's explicit information about the *task* to take a sequence of actions towards the goal. This combination of generating (and updating) a distribution of possible maps, then planning a sequence of belief space actions using a learned policy enables our approach to follow complex natural language directions through unstructured unknown environments.

For comparison, the result of the basic policy (without hypothesizing maps or



(a) The simulated robot once again starts at the blue square. The policy immediately uses the door right in front of the robot for the SDC "go to the door," without looking for any water fountains.

(b) Without using the navigation information, the policy decides to stop too early (note there are two doors in the same hallway). It then transitions to the second SDC, "turn right."

(c) When a right turn is available, the policy decides to take it even though it leads into a bathroom.

(d) The policy is now on the third SDC ("go straight to the cabinet"). The toilet object in the environment is semantically likely to co-occur with a "cabinet," so the policy stops following the direction here.

Figure 6.5: Sequence of policy decisions without using navigational information for the same direction as above. Because the policy does not use the implicit information about the (correct) door's location, it stop too early on the first SDC (at an incorrect door). This causes the error to compound when it turns right into the bathroom. The color-coding is the same as in Figure 6.4, but in this case the policy does not generate a distribution of semantic maps from the language, nor does it use the additional navigation information contained in the language.

using the additional information in the direction) on the same direction is illustrated in Figure 6.5. In this case the policy turns right too early on the first SDC, because it did not utilize the additional information about the location of the door (the direction is ambiguous without this). The policy then turns right into a bathroom, and completes following the direction there (14.7 m from the correct destination).

Cross-validation results

We now evaluate the policy quantitatively over many repeated cross-validation trials on the extended corpus, by training a policy on about half of the directions, and testing it on the remaining held out directions. As with our previous approach, we will measure for each trial the average ending distance across the held out directions, and the success rate for various distance thresholds. These results are on the combined set of 55 directions (40 basic directions and 15 complex directions), treated as one large corpus (we did not sample the complex directions separately). We ran 200 trials of cross-validation, using 28 directions for training and 27 for testing.

As expected, the results show that our belief space policy performs significantly better than the previous approach on this corpus. As with the single validation example in Figure 6.4, in this setting the semantic infers a distribution of maps consistent with the additional information contained in the language. For instance, the average ending distance error of the test directions across all cross-validation trials, shown in Figure 6.6, demonstrate that reasoning about a distribution of landmarks (and planning in the resulting belief space) reduces the average ending distance error significantly. By hypothesizing an initial distribution of landmarks based on the command and then utilizing any navigation landmarks in the command (after they are detected), the policy is able to improve its decision making. Similarly, the success rate when inferring a distribution of landmarks improves significantly. The results for the average success rate across all trials is better with belief space reasoning than without (Figure 6.7). This is true for every possible distance threshold (i.e., changing how far away the robot must be from the true goal to declare the trial a success).

6.4 Chapter Summary

This chapter extended our sequential decision making approach to following natural language directions from Chapter 3 to utilize the additional information provided by the semantic map inference (namely, a distribution of possible maps). The samples (particles) from this map distribution contain the possible location of landmarks inferred from the natural language command and robot's sensor stream. While this information can guide the robot's decision making, the robot must now reason in the

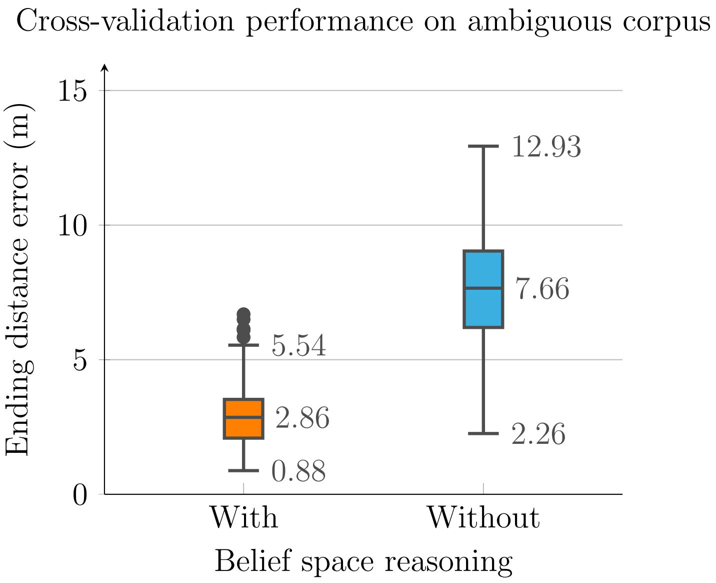


Figure 6.6: Reasoning about the distribution of landmarks (with belief space reasoning) reduces the average ending distance error across 200 cross-validation trials, compared to without belief space reasoning. Lower error is better.

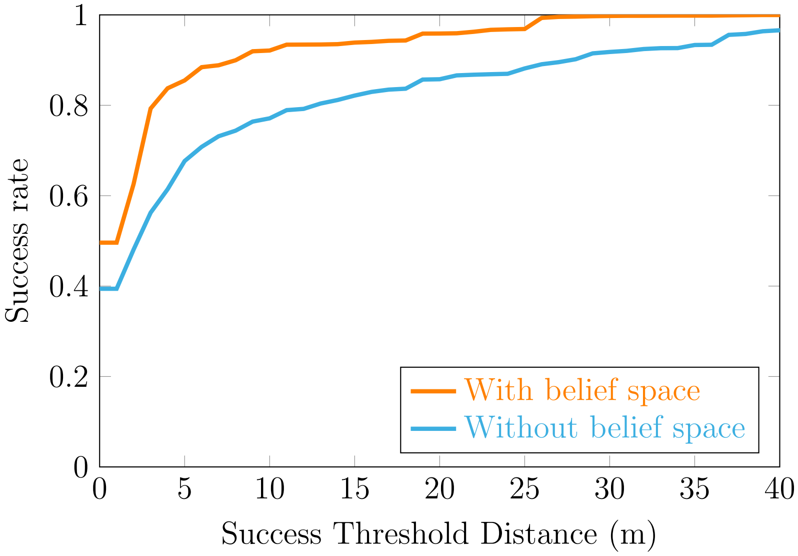


Figure 6.7: Reasoning about landmark distributions improves the average success rate of the system across all distance thresholds. Each curve represents the average success rate across 200 cross-validation trials for any given distance threshold. Higher success rate is better.

belief space of possible environments.

Our framework for following directions remains unchanged: a policy will enumerate a set of feasible actions from its current state, evaluate the cost of each action, and execute on the robot the one with the lowest cost. This will generate a new observation of the world, and update its map representation. The main difference is that the world representation is now a distribution over the valid semantic maps (the combined metric, topological, and semantic representation of the world). This map distribution contains hypotheses of landmark locations that are inferred based on observations from the robot's sensor *and* the natural language command, effectively treating language as another sensor.

The challenge remained to effectively use this additional information to successfully inform the policy. We formulated a belief space policy that computes the cost of each action using a kernel distribution embedding of the action feature distribution (induced by a distribution over possible landmarks). Our belief space planning problem is distinct from others in that our primary objective is to follow the direction, not take actions that optimize some metric on the belief (e.g., reducing uncertainty). The policy formulation enables us to do this efficiently: the additional cost is simply computing more features (determined by the number of belief samples).

We introduced a method to train the policy using a simple extension of our imitation learning formulation from Chapter 4. The resulting policy encapsulate the expert's knowledge of the uncertainty in the environment, and leads to effective decision making in a distribution of maps. We showed that this approach is effective at following natural language directions, outperforming our prior approach when the directions are complex. These complex directions contain references to navigation landmarks, intended to resolve between multiple possibilities (for example, "go to the door across from the whiteboard"). Our learned belief space policy resulted in significantly lower error in the average ending distance, and significantly higher success rates.

The next two chapters will demonstrate this approach on several integrated robot platforms. As we will show in the next chapter, our approach of inferring maps for following natural language directions leads to performance that approaches the performance of the robot operating with a completely known map.

Chapter 7

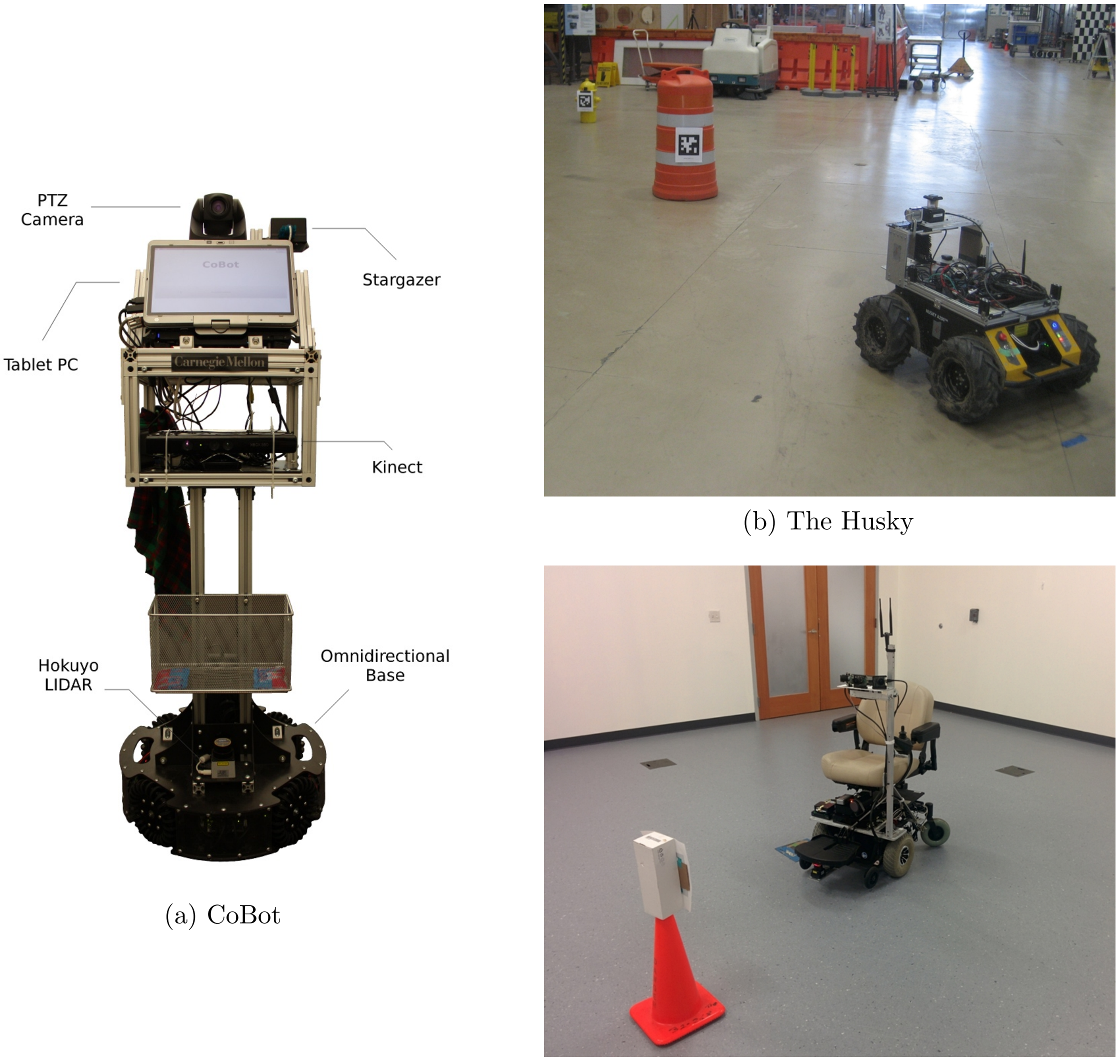
Integrated Demonstrations on Autonomous Indoor Robots

> I urge you to please notice when you are happy, and exclaim or murmur or think at some point, 'If this isn't nice, I don't know what is.'
>
> Kurt Vonnegut

We now present results of applying our approach of using a policy to follow natural language directions through unknown environments on three robots that operate in unstructured indoor environments:

- **CoBot**, an autonomous robot developed at CMU that performs service tasks and interacts with people in an indoor office environment [125, 164].
- **The Husky**, an all-terrain mobile robot that operates indoor and outdoor. This robot was equipped with a sensor package and autonomous navigation capabilities by a consortium as part of the RCTA project [111].
- **The MIT intelligent wheelchair**, an autonomous voice-commandable wheelchair developed to assist users in a variety of tasks [55].

Each of these platforms (shown in Figure 7.1) has different sensor configurations, low-level planners, and ways of mapping the world. However, as we will show, our



(a) CoBot

(b) The Husky

(c) Autonomous Wheelchair

Figure 7.1: Our approach to following natural language directions generalizes across a wide variety of platforms and environments, shown here. CoBot (7.1a) operates primarily in indoor office environments, the Husky (7.1b) operates primarily in larger open areas, and the autonomous wheelchair (7.1c) operates in both types of environments.

approach is general and can therefore apply across this wide variety of robots and environments.

In most cases we present anecdotal results on each robot, testing the complete integrated system on a small number of runs for different commands. We will also present more extensive results of the same integrated system running in simulation. For more comprehensive results evaluating our approach to learning policies (isolated from the entire integrated system), see the results in Chapters 4 and 6.

We begin in Section 7.1 with a generalization of our approach from Chapters 3 and 4 to an entirely novel environment by applying our learned policy (trained on a corpus of directions in the MIT Stata center) to the CoBot robot operating in CMU's Gates building, an environment the policy had never seen. These experiments are joint work with Thomas Kollar. In Section 7.2, we then apply our approach to inferring semantic maps (presented in Chapter 5) on the Husky, commanding the robot with relatively simple natural language directions. We show that inferring a distribution of maps improves performance. In Section 7.3, we then combine this semantic map inference approach with a learned belief space policy (introduced in Chapter 6), where the commands are more complex and contain ambiguity, on the autonomous wheelchair. The results presented in Sections 7.2 and 7.3 are adapted from two prior publications (Duvallet et al. [42], Hemachandra et al. [57]); these are the result of joint work with Sachithra Hemachandra, Thomas Howard, and Matthew Walter.

In addition to showing successful implementations of our approach on three different mobile robots, the results of our experiments support the following conclusions:

- Real robots can follow natural language directions through unknown environments by framing the problem as sequential decision making under uncertainty.
- Policies trained using imitation learning encapsulate the uncertainty present in the unknown parts of the environment and effectively capture the expert's knowledge.
- Our feature representation enables generalization to new commands in environments the robot has never encountered.
- Inferring a distribution of maps (using the information implicitly specified in the language) improves the performance of the robot, approaching the level of

a robot operating with a fully-known map.

- Our integrated approach can recover from false initial assumptions (e.g., in cases of multiple possibilities) to successfully follow ambiguous directions.

These findings support the thesis statement introduced in Chapter 1: following natural language directions through unstructured unknown environments can be formulated as sequential decision making under uncertainty, furthermore, language can be used as a sensor to infer maps that improve the ability of the robot to follow directions. These experiments are one step towards demonstrating effective human-robot coordination in shared teams that communicate using natural language.

7.1 Generalization to Novel Environments on CoBot

The first integrated robot experiment we present demonstrates the ability of our policy to generalize to a novel environment. More specifically, we train the policy on directions in one building, and evaluate it on directions in a *different* building. This demonstrates the ability of the policy to make a sequence of decisions in an unknown environment it has never observed. In this section we evaluate the basic policy formulation (without belief space reasoning) introduced in Chapters 3 and 4: we provide the robot with a natural language command and no map of the world, then evaluate its actions.

Methods

We demonstrate our ability to generalize to novel environments using CoBot, a research platform developed at CMU by Veloso et al. [164]. This robot is a mobile indoor service robot that can be used for a variety of tasks: picking up and delivering packages, escorting visitors to meetings, and remote telepresence. Users can request and schedule tasks via a website. A team of several CoBot robots has been in operation since 2009 in multiple buildings on CMU's campus: Wean Hall, Gates Hall, and Newell-Simon Hall, and these robots have collectively traveled over 1000 km [16].

Our approach to handling mapping and perception on this robot is similar to the results presented in Section 4.5: we use a fully-annotated semantic map that an external process incrementally "reveals" to the robot as it travels through the world (obeying visibility constraints). This is possible since CoBot normally operates in a fully-known map that includes a metric map of the building, a topological graph of the hallways, and the location of semantically annotated landmarks (such as elevators, kitchens, printers, etc.). Our focus in this section is on the problem of generalizing to different environments the policy has never before seen.

We trained the policy using 30 directions through one floor of MIT's Stata center (a subset of the corpus for the results in Section 4.5). After training, the only

We gratefully acknowledge Joydeep Biswas, Brian Coltin, and Manuela Veloso for their help running the experiments presented on CoBot.

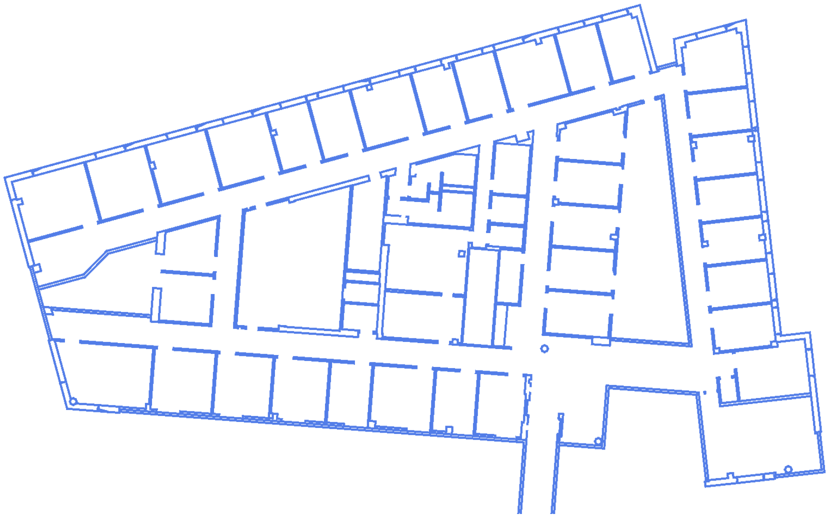


Figure 7.2: Floor plan of the Gates-Hillman Center at CMU.

information we retained were the learned weights in Equation (3.10). These weights specify how each feature (for a state-action pair) affects the policy's cost function. We then commanded CoBot with a variety of natural language commands in CMU's Gates Hall (shown in Figure 7.2). Note that the policy has never seen a single demonstration from this new environment, all of its training data came from a corpus of directions through one floor of a building at MIT. While these are both indoor office-like environments, the specific environment structure is different, the semantic objects are in different configurations (and have different shapes), and even the units used in the map are different. We computed features for actions in this environment using the same code.

Results

We only present qualitative results of following natural language directions in a novel environment using CoBot. This is because we ran a relatively small number of commands, and the language in the commands was also quite simple. In addition to serving as a real world validation of our approach, these results demonstrate a very important finding of our research: the learned policy can generalize to environments it has never encountered before.

A trace of CoBot correctly following the natural language direction "Go past

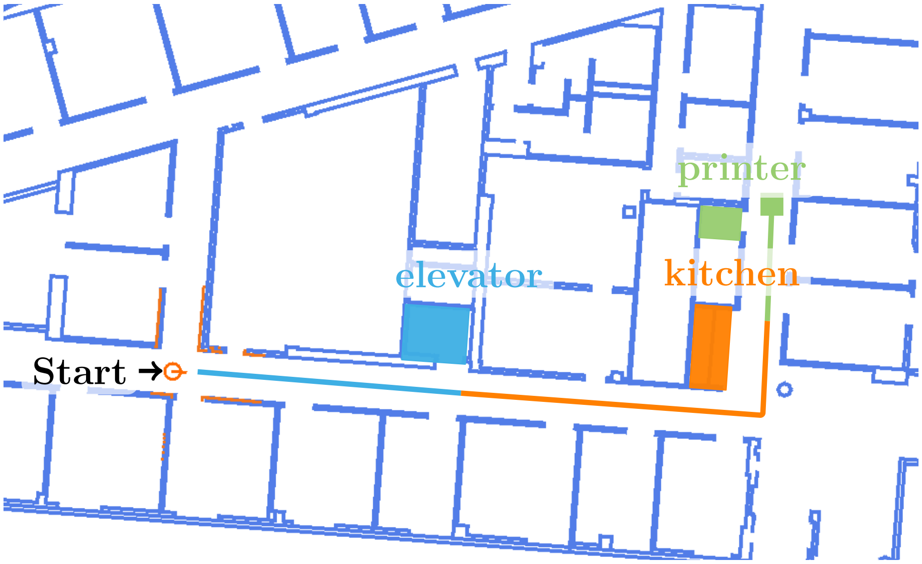


Figure 7.3: Trace of successful CoBot run for following the command "go towards the elevator, turn left towards the kitchen, and go towards the printer." We show the complete robot's path through the world (color-coded by SDC), as well as the landmarks the policy used to follow the direction. Note that we show the entire environment, but CoBot only had access to the parts of the environment it had visited (a partial map).

the elevator, turn left towards the kitchen, and go towards the printer" is shown in Figure 7.3. After it detects the elevator object, the robot continues taking a sequence of actions until it determines it is "past" the elevator. The robot then transitions to the second SDC, and even though it cannot yet see the kitchen or a left turn it continues down the same hallway. CoBot sees the left turn and the kitchen when it arrives in a common area, and correctly takes a left turn. After taking actions for the third SDC, the policy declares when it believes it has reached the destination and stops at the printer.

For the command "Go past the bathroom, turn left, and turn left to the elevator" (shown in Figure 7.4), CoBot initially correctly follows the first SDC and transitions at the appropriate time. However, the policy chooses to continue past the first left turn (incorrectly). When CoBot reaches the end of the hallway, the policy determines it has made a mistake and the robot backtracks to the correct hallway. In this situation, the policy's cost for the stop action was higher than the cost of backtracking to the hallway. After traveling down the hallway, it correctly calls the stop action,

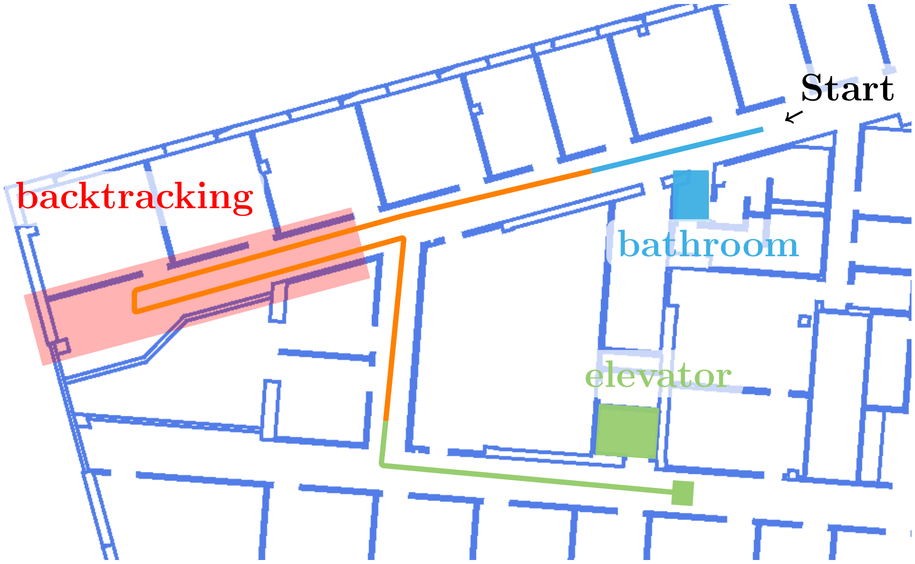


Figure 7.4: Trace of a run where CoBot made an error and backtracked. The command was "go past the bathroom, turn left, and turn left to the elevator". In this case, CoBot did not turn left correctly, but realized it made a mistake when it reached a dead end. It then backtracked and continued along the rest of the direction correctly.

transitions to the final SDC (turning left to the elevator), and reaches the elevator (overshooting slightly).

Discussion

Using CoBot, we were able to demonstrate the generalizability of our approach to entirely novel environments (i.e., one the robot policy had never seen). The ability to generalize to novel environments is especially important for robotics because training policies on each environment separately is costly and in some cases may be infeasible. Additionally, we demonstrated that framing direction following as sequential decision making can enable real robots to follow natural language directions through unknown environments, which supports our thesis statement. This robot experiment validates our approach to training a policy in an unknown environment, which enables the policy to reason about the unknown pats of the environment. We also demonstrated an example of the robot making a mistake and recovering, which was possible because the policy had demonstrations of both successes (what the expert did) and failures (what the expert would have done to recover) from the training data.

These anecdotal experiments are one step towards showing that our approach generalizes well across different environments. Combined with the fact that our approach does not require a map of the environment ahead of time, this holds the promise of enabling robots to understand natural language commands in new environments without requiring time-consuming environment-dependent training. The ability for robots to operate successfully in novel unknown environments will improve their ability to collaborate with people and be effective partners in human-robot teams.

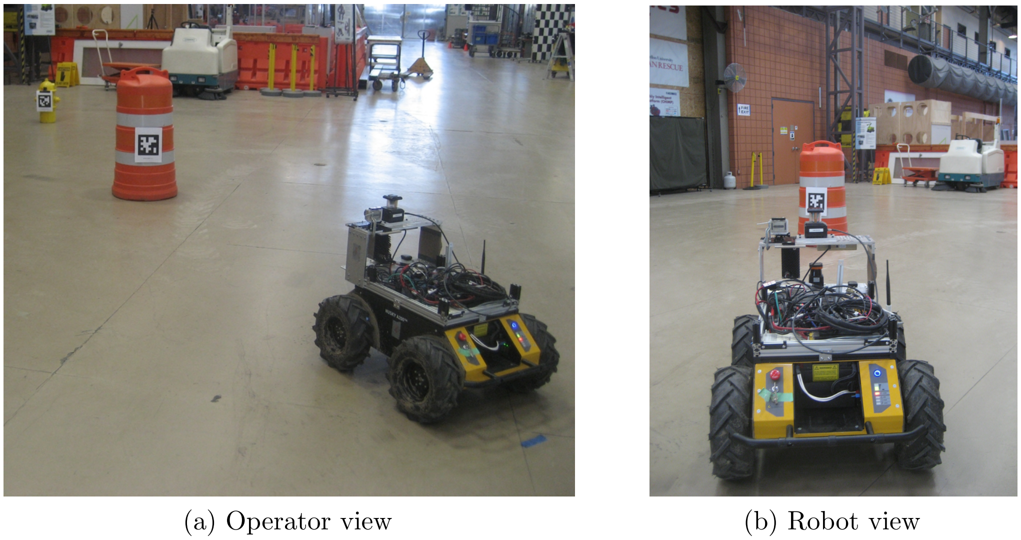
7.2 Demonstration of Semantic Map Inference on the Husky Robot

The previous experiments on CoBot demonstrated our sequential decision making formulation on a robot in a novel unknown environment. We now apply the language-driven semantic map inference approach presented in Chapter 5 to the Husky robot platform, and demonstrate that inferring maps from a natural language direction improves the performance of the robot when it is following the direction. Consider the environment in Figure 7.5, showing both the operator's and Husky's viewpoints. If the operator (who can clearly see both the cone and the hydrant) instructs the robot (that can only sense a cone) to "drive to the hydrant behind the cone," our map inference approach will leverage the information in the language (namely, there is a hydrant and it is located behind the cone) to infer a suitable distribution of environments. The policy will then use this inferred map distribution to make a sequence of belief space decisions.

Methods

The following experiments measure the ability of the robot to execute the intended command. We will compare our approach with an inferred map distribution (using information contained in the language) against a baseline of using a complete map of the environment (acquired by manually driving the robot around the environment). The Husky is running the Semantic Map inference framework described in Chapter 5, with a graphical user interface to input an unstructured natural language command. The robot is equipped with an Adonis camera, and in these experiments we use the AprilTag fiducials [64, 112] for object detection and localization. This once again simplifies the perception problem while observing all physical constraints (namely, sensing range and line of sight). In these first experiments, the robot's belief space policy cost function is hand-tuned to minimize the expected distance to the goal landmark distribution. In Section 7.3 we will use a belief space policy learned from demonstrations.

The following experiments on the Husky were possible because of help from the greatest Husky wranglers of all time, Jean Oh and Bob Dean.



(a) Operator view

(b) Robot view

Figure 7.5: Differing viewpoints lead to mismatched world representations. The operator can clearly see both the hydrant and the cone, but the robot can only see the latter. Our approach leverages the information contained in a natural language command to narrow this gap by inferring possible locations of the hydrant.

In this experiment we performed 24 trials by commanding the robot to navigate to the (initially unknown) hydrant. We varied both the starting environment model (known vs. unknown) and the command (with and without a relation): the two natural language commands were “go to the hydrant behind the cone” and “go to the hydrant.” Each condition (starting map and command) consisted of six trials. This allowed us to measure the ability of the robot to use the additional information from the spatial relation in the command, as well as compare our approach to that of knowing the complete map ahead of time. We consider a trial to be a success if the robot stops within 1.5 m of the hydrant, and a failure if the robot stops or the operator must take over (e.g., because of some unsafe action).

Results

When the robot had access to a complete map of the environment, it successfully executed 100 % of the commands (whether or not the language contained information about the hydrant’s location). When the robot started with an unknown map and the direction included the spatial relation (“behind the cone”), our approach successfully

Table 7.1: Experimental results on the Husky for the command "go to the hydrant behind the cone." Both conditions (known and unknown map) represent six unique trials. We report mean time and distance (with 95 % confidence intervals) and success rates for the different conditions. As we can see, our approach of inferring a map distribution and using a belief space policy yields performance in unknown environments that approaches having a complete map a priori.

Map	Distance (m)	Time (sec)	Success (%)
Known	8.4 ± 1.3	26.4 ± 4.2	100.0
Unknown	8.1 ± 0.6	34.0 ± 18.6	83.3

executed the command 83.3 % of the time, even though the goal object (the hydrant) was not initially visible or known in advance (Table 7.1). When the robot was given a command without any spatial relation ("go to the hydrant") in an unknown map, the inferred map distribution was not sufficient to help the robot find the hydrant (0 % success rate). (A better uninformed search behavior could have helped the robot eventually find the hydrant, but we did not implement this.)

As we expected, the results indicate that inferring a map distribution helps the robot follow the command correctly. Additionally, the distance traveled by the robot using an inferred map distribution is similar to the distance traveled when the robot has access to a complete world map. The main difference was in the time taken by the robot to follow the direction (about 29 % more for the inferred maps than the fully-known map), due to the time required to integrate new sensor measurements, update the distribution of inferred maps, and evaluate the cost of possible actions. The approach overview diagram (Figure 5.2) illustrates data collected during one run on the Husky for the command "go to the hydrant behind the cone," where the robot starts with an unknown map but can observe the cone.

Discussion

These experiments suggest it is possible to achieve a level of performance in an unknown environment that is similar to knowing the map a priori by exploiting the information implicitly contained in the command. Using the user's natural language instruction, we can infer the relationship between objects that may not be initially observable without having to consider those annotations as a separate utterance. By learning and reasoning about a distribution over the possible maps and behaviors, we can solve for a policy that explicitly takes into account the uncertainty in the map. The next experiments on the autonomous wheelchair will integrate this map inference approach with a learned belief space policy, and demonstrate this approach on more complex environments and directions.

7.3 Demonstration of Belief Space Policy on the Autonomous Wheelchair

The previous experiments on the Husky platform demonstrated the effectiveness of our approach to following natural language directions in unknown environments by hypothesizing the environment beyond the robot's sensor range, using the information contained in the language. As we discussed in Chapter 6, the policy must now reason in the belief space of possible maps. We now present results of our approach in more complex environments (and with more intricate commands) that highlight the ability of the policy to reason about a distribution of landmarks, deal with ambiguity, and recover from false assumptions.

We ran these experiments on a voice-commandable wheelchair located at MIT, shown in Figure 7.1c. This robot has three forward-facing cameras with a collective field-of-view of 120 degrees, and both forward- and rear- facing LIDARs. We again used AprilTag fiducials [64, 112] to detect and estimate the relative pose of objects in the environment, subject to self-imposed sensing range restrictions. The use of this additional platform further demonstrates the applicability of our algorithm to various mobile robots with different platform configurations, underlying motion planners, and sensor configurations. We begin by presenting experiments that require reasoning about objects in an open environment, extending those from Section 7.2. We then demonstrate our approach on directions through an office environment that require reasoning about regions, such as hallways and kitchens.

Reasoning about objects in an open environment

The first set of experiments on the wheelchair extend the ones from the Husky by using more complex environments and commands. All the commands direct the robot to an initially-unknown landmark (in this case a hydrant), and involve different spatial relationships to cones ("behind," "nearest,"), or no relation at all (i.e., "go to the hydrant"). Some environments were ambiguous: there were multiple cones the hydrant could be behind, or there were multiple hydrants and the robot had to select the one "nearest" to the cone.

In each experiment, a human operator issues one of the following natural language

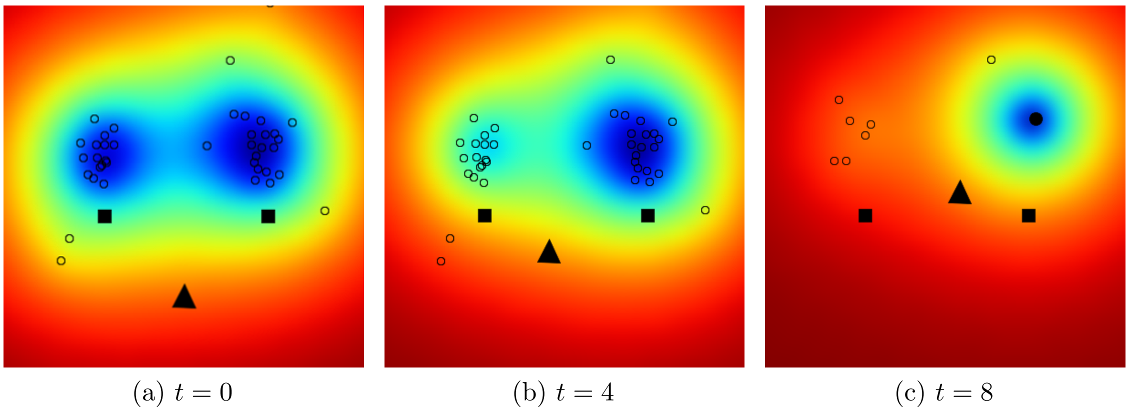
Table 7.2: Experimental results on the wheelchair in an open environment, showing the mean and standard deviation across trials for various conditions (ten runs for our approach and one run with a known map). Each row is a different world configuration (Hydrant, Cone), sensing range, and spatial relation in the natural language command. One of the hydrants was the goal each time.

World	Sensing Range (m)	Spatial Relation	Success Rate (%)		Distance (m)	
			Known	Ours	Known	Ours
1H, 1C	2.5	null	100	100	4.7	16.6 ± 7.2
1H, 1C	2.5	“behind”	100	100	4.7	9.9 ± 3.4
1H, 2C	3.0	“behind”	100	100	4.6	7.6 ± 2.1
2H, 1C	2.5	“behind”	100	80	5.3	6.0 ± 1.4
2H, 1C	4.0	“nearest”	100	100	4.1	5.0 ± 0.4
2H, 1C	3.0	“nearest”	100	50	6.3	7.1 ± 0.6

commands to the robot: “go to the hydrant,” “go to the hydrant behind the cone,” or “go to the hydrant nearest to the cone.” For each of these commands, we consider different environments by varying the number and position of the cones and hydrants (the room was otherwise free of obstacles), and by changing the robot’s sensing range. For each configuration of the environment, command, and sensing range, we performed ten trials with our algorithm. For a ground truth baseline, we performed an additional trial with a completely known world model. We consider a run to be successful if the robot’s final destination is within 1.5 m of the intended goal.

As in our simpler experiments on the Husky, our results on the Wheelchair (shown in Table 7.2) demonstrate the utility of utilizing the extra information contained in the language command to infer a distribution of maps, and using this distribution as input to a belief space policy. We find that our algorithm is able to take advantage of available relations (“go to the hydrant *behind the cone*”) to yield behaviors closer to that of ground truth, compared to the command that does not provide a relation (“go to the hydrant”). The distance traveled in the first two rows of Table 7.2 shows a sharp decrease when using the additional information in the language.

When there is ambiguity in the command, for example the command “go to the


(a) $t = 0$

(b) $t = 4$

(c) $t = 8$

Figure 7.6: Visualization of the landmark distribution over time for the command "go to the hydrant behind the cone," where the triangle denotes the robot, squares denote observed cones, and circles denote hydrants that are either sampled (empty) and observed (filled). Figure 7.6a: The robot starts off observing both cones, and hypothesizes possible hydrants that are consistent with the command. Figure 7.6b: the robot moves towards the left cluster, but after not observing the hydrant, the map distribution peaks at the right cluster. Figure 7.6c: The robot then moves right and observes the actual hydrant. This visualization is from a simulation run (for illustration), but the robot experiments result in very similar distributions.

hydrant behind the cone" in a world with two cones in the environment (third row of Table 7.2), the robot does not initially know *which* cone the hydrant is behind. In this case the robot generates a bi-modal distribution of maps: some of the semantic maps contain a hydrant behind the first cone, others contain a hydrant behind the second cone. This is illustrated in Figure 7.6, where the policy begins with roughly equal probability mass behind each of the two cones. When the robot travels behind a cone and does not see a hydrant, the semantic map distribution down-weights those particles. The policy uses this updated map distribution to select a new action, and visits the area behind the other cone. After the robot detects the hydrant, the map distribution converges to the true environment and the policy eventually declares it is done following the direction. This explains our result on the robot: the robot successfully recovers from any false assumptions (100 % success rate), but travels slightly longer than the fully-known map case on average (7.64 m vs 4.58 m). This is because in some of the trials, the robot goes behind the incorrect cone and recovers.

When the environment contained two hydrants (last three rows of Table 7.2), the robot trials exhibited an interesting failure mode that we did not expect. If the robot detects one of the two hydrants before a cone, the semantic map distribution then hypothesizes the existence of cones around the hydrant (using the relation in the command). This leads to a behavior distribution peaked around the detected hydrant, and the policy goes towards it without looking for the possibility of another hydrant in the environment. This effect was most pronounced with shorter sensing ranges (e.g., a 3 m sensing range for the command "go to the hydrant nearest to the cone" resulted in the robot reaching the goal in only half of the trials compared to a 4 m sensing range). Adding a simple behavior that checks for the existence of *all* necessary landmarks for the direction (both the cone and hydrant) could fix this behavior.

These results demonstrate the usefulness of utilizing all of the information contained in the instruction, such as the relation between various landmarks in the environment that can be helpful during navigation. We now describe experiments that go beyond simple landmark-based commands and describe real world navigation commands through indoor office environments that involve reasoning about previously unknown regions such as kitchens and hallways.

Reasoning about regions in an office environment

In these experiments we go beyond object-based directions to command the robot with directions that refer to human-centric *regions* in an indoor office environment (for example, "go to the kitchen that is down the hallway"). This requires the ability to detect and semantically label regions in the environment (e.g., kitchens, hallways), reasoning about spatial relationships between them when hypothesizing possible semantic maps, and following more complex natural language directions that include more verbs and relationships to the goal region (e.g., "past the kitchen"). In this experiment the wheelchair is operating on a floor of MIT's Stata Center containing several hallways, offices, lab spaces, and a kitchen on the same floor. We used the automatic region segmentation presented by Hemachandra et al. [56], but placed AprilTag fiducials [112] to identify the region type once the robot enters it. The wheelchair starts with no map of the environment, and we directed it to execute the instruction "go to the kitchen that is down the hallway."

Table 7.3: Experimental results on the MIT autonomous wheelchair in an office environment. The command was “go to the kitchen that is down the hallway.” This table shows the mean and standard deviation of the distance traveled and time to reach the goal for each experimental condition: known map (the robot starts with a complete map of the environment), with language (the robot uses the implicit information in the command), and no language (the robot does not use the implicit information). Again, we see that utilizing the information in the language (our approach, **With Language**) enables the robot to travel about the same distance as the known map case.

Algorithm	Distance (m)	Time (s)
Known Map	13.1 ± 0.7	62.5 ± 16.6
With Language	12.6 ± 0.6	122.1 ± 32.5
No Language	24.9 ± 13.6	210.4 ± 97.7

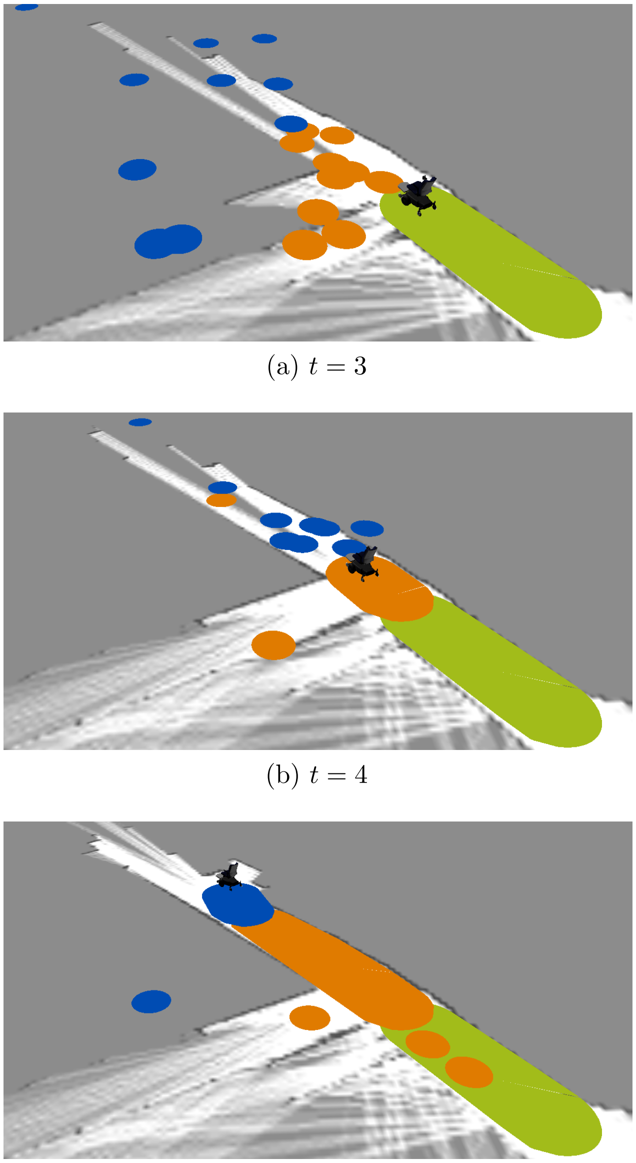
We compare our framework of inferring semantic maps using the natural language command against two other methods. One uses a *known* map of the environment in order to infer the actions consistent with the route direction. The second method assumes no prior knowledge of the environment (as with ours) but does not use language to modify the map (it only searches for the goal). We performed six runs with our algorithm, three runs with the known map method, and five with the method that does not use language, all of which were successful.

The results of this experiment (measuring the distance traveled and total execution time) once again show that our approach of planning in an inferred map distribution enables performance that is close to operating in a fully-know map (Table 7.3). The robot traveled about the same amount in our approach and when it had access to a completely known map a priori, because it leveraged the knowledge about the location of the kitchen contained in the natural language instruction. The semantic map distribution reflected this knowledge by updating the hypothesized locations of possible kitchens in the map particles once it detected the hallway (we show the semantic map evolution over time in Figure 7.7). The robot took a longer time to execute the command because it had to take a sequence of small actions, updating its map distribution and solving the belief space policy after each action. By comparison, the known map condition traveled directly to the kitchen in a single action. The full map of the environment, shown in Figure 7.8, was collected by driving the robot in the environment manually. The approach that did not use language to update its map distribution explored the environment more or less randomly until it located the kitchen; as expected this resulted in poor performance according to both metrics.

Discussion

These experiments on the autonomous wheelchair along with those on the Husky demonstrate the effectiveness of our approach of inferring a map distribution (using the information contained in the language) and then using a belief space policy to take a sequence of actions towards the goal. As with the rest of our work, this relies on a formulation of natural language direction following as sequential decision making. Effectively, we use language as a sensor that can hypothesize the parts of the environment that are beyond the robot's sensor range. The main difference is that the policy must reason in the belief space of possible maps, since we infer a distribution of possible environments. As the robot gains more information about the world, it updates the map distribution before taking its next action. We have shown that our approach results in performance in unknown environments that is close to the performance of the robot operating in a fully-known map, simply by leveraging the additional information that is implicitly contained in the language. In cases where the language is ambiguous, we showed that our approach is able to recover from initial false assumptions to follow the direction successfully.

This ability to reason about the parts of the environment that have not yet been observed but are mentioned in a natural language direction is one important step towards seamless human-robot interaction with lay users in environments that are unknown to the robot. In addition, this explicit representation of the robot's belief (as a distribution of maps) could become a useful visualization tool by explicitly representing the robot's knowledge of the world in a human-understandable form.



(a) $t = 3$

(b) $t = 4$

(c) $t = 8$

Figure 7.7: Visualization the evolution of semantic maps for one run on the robot, for the command "go to the kitchen that is down the hallway." Sampled regions are drawn as small circles and visited regions are shown with the area filled in (lab: green, hallway: orange, kitchen: blue). The robot first samples possible locations of the kitchen and moves towards them (7.7a), then observes the hallway and refines its estimate using the "down" relation provided by the user (7.7b). Finally, the robot reaches the actual kitchen (7.7c) and declares it has finished following the direction. For comparison the ground truth environment is shown in Figure 7.8.

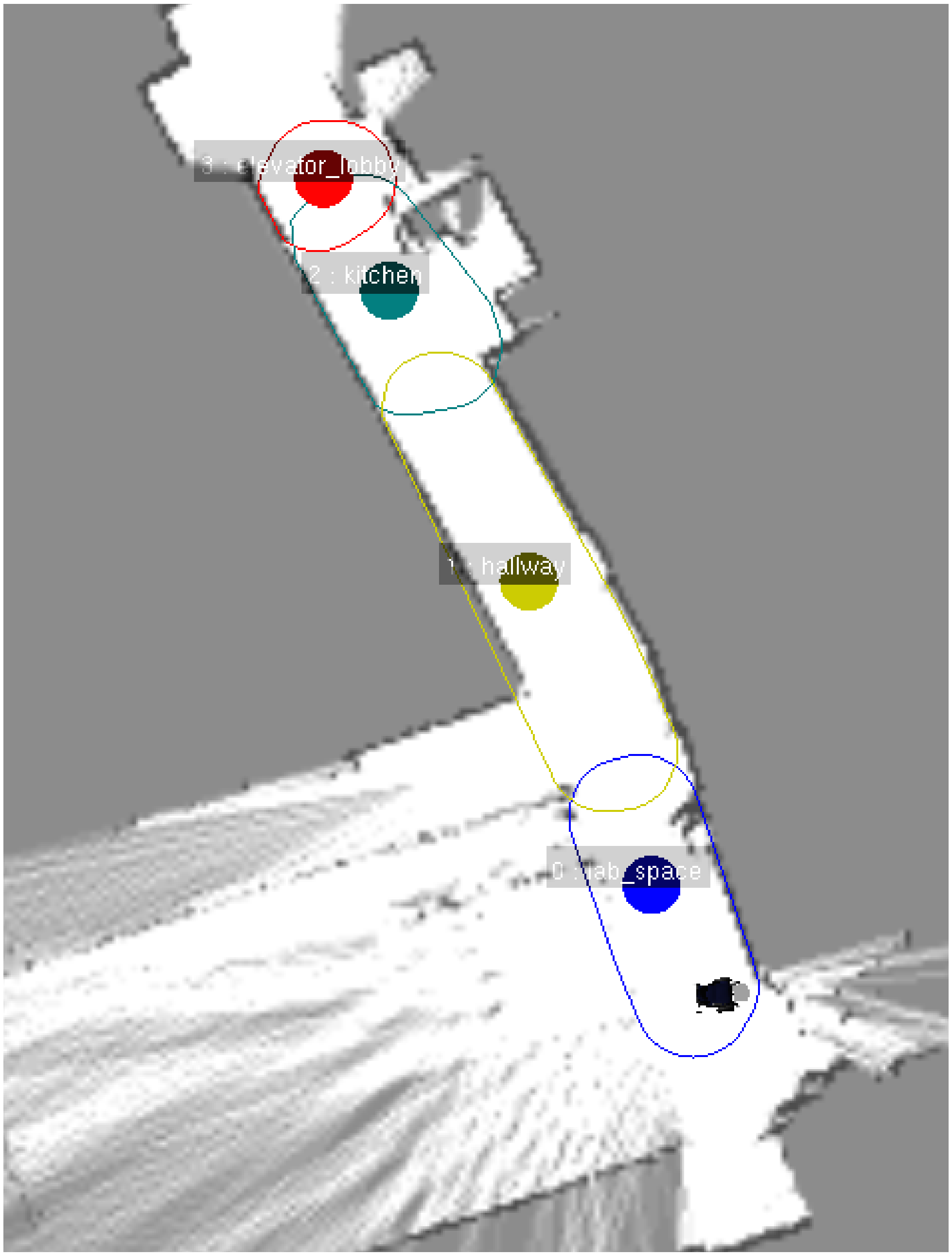


Figure 7.8: Full map of environment collected by driving the robot manually around the environment. The regions are lab space (blue), hallway (yellow), kitchen (green), and elevator lobby (red).

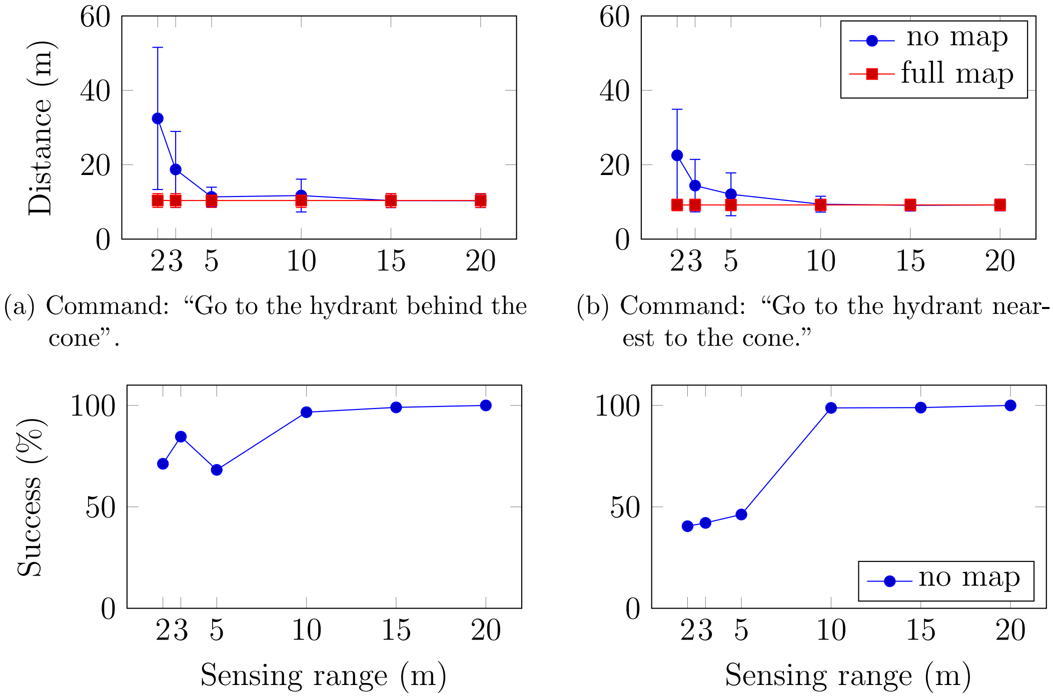
Table 7.4: Direction following performance in simulation using belief space policy: mean and standard deviation across trials for various conditions. Each row is a different world configuration (Hydrant, Cone), sensing range, and spatial relation in the natural language command. One of the hydrants was the goal each time.

World	Sensing Range (m)	Spatial Relation	Success Rate (%)		Distance (m)	
			Known	Ours	Known	Ours
1H, 1C	3.0	null	100.0	93.9	8.8 ± 1.7	16.8 ± 7.9
1H, 1C	3.0	"behind"	100.0	98.3	8.8 ± 1.7	13.4 ± 7.0
1H, 2C	3.0	null	100.0	100.0	11.2 ± 1.4	32.5 ± 18.5
1H, 2C	3.0	"behind"	100.0	99.5	11.2 ± 1.4	40.0 ± 29.7
2H, 1C	3.0	null	100.0	54.4	10.5 ± 1.8	21.6 ± 10.3
2H, 1C	3.0	"behind"	100.0	67.4	10.4 ± 1.8	18.7 ± 10.2
2H, 1C	5.0	"nearest"	100.0	46.2	9.2 ± 1.5	12.1 ± 5.8

7.4 Simulated Belief Space Experiments with Parameter Variation

Next, we evaluate the entire framework through an extended set of simulations in order to understand how the performance varies with the environment configuration and the command. We consider four environment templates, with different numbers of hydrants and cones. For each configuration, we sample ten environments, each with different object poses. For these environments, we issued three natural language instructions "go to the hydrant," "go to the hydrant behind the cone," and "go to the hydrant nearest to the cone." We note that these commands were not part of the corpus that we used to train the DCG model. Additionally, we considered six different settings for the robot's sensing range (2 m, 3 m, 5 m, 10 m, 15 m, and 20 m) and performed approximately 100 simulations for each combination of environment, command, and range. As a ground truth baseline, we performed ten runs of each configuration with a completely known world model.

Our method's success rate and distance traveled by the robot for these 100 simulation configurations is shown in Table 7.4. We considered a run to be successful



(a) Command: “Go to the hydrant behind the cone”.

(b) Command: “Go to the hydrant nearest to the cone.”

(c) Command: “Go to the hydrant behind the cone”.

(d) Command: “Go to the hydrant nearest to the cone.”

Figure 7.9: Distance traveled and success rate and for our approach in two simulated scenarios, comparing our approach with no prior map against a completely known prior map. With increasing sensor range, success rate improves and the distance traveled approaches that of the full prior map.

Table 7.5: Direction following performance in simulation using belief space policy (mean and standard deviation across 10 trials), for the command "go to the kitchen that is down the hallway."

Algorithm	Distance (m)	Time (s)
Known Map	12.9 ± 0.1	18.3 ± 3.5
With Language	16.6 ± 6.9	82.8 ± 10.6
Without Language	25.3 ± 13.0	85.6 ± 17.8

if the planner stops within 1.5 m of the intended goal. Comparing against commands that do not provide a relation (i.e., "go to the hydrant"), the results demonstrate that our algorithm achieves greater success and yields more efficient paths by taking advantage of relations in the command (i.e., "go to the hydrant behind the cone"). This is apparent in environments consisting of a single figure (hydrant) as well as more ambiguous environments that consist of two figures. Particularly telling is the variation in performance as a result of different sensing range. Figure 7.9 shows how success rate increases and distance traveled decreases as the robot's sensing range increases, quickly approaching the performance of the system when it begins with a completely known map of the environment.

The same interesting failure case occurs in simulation when the robot is instructed to "go to the hydrant nearest to the cone" in an environment with two hydrants. In instances where the robot sees a hydrant first, it hypothesizes the location of the cone, and then identifies the observed hydrants and hypothesized cones as being consistent with the command. Since the robot never actually confirms the existence of the cone in the real world, this results in the incorrect hydrant being labeled as the goal.

For the command "go to the kitchen that is down the hallway," we again evaluated our method in simulation. We carried out the same comparison in a simulated world comprised of an office, hallway and a kitchen (with the robot starting off in the office). We achieved similar results to our real world experiments; our method achieved comparable results to the known map method while outperforming the method without language (Table 7.5). Each trial represents ten runs.

7.5 Discussion

First, we demonstrated our approach of using a policy to follow natural language directions through an unknown environments on CoBot, an autonomous platform that has demonstrated impressive long-term deployments and interacts with users on a regular basis. While these anecdotal experiments did not address many important issues such as perception and online mapping, it did demonstrate that sequential decision making is successful at following natural language directions, and most importantly that our approach can generalize to novel environments the robot had never seen. We used imitation learning to train a policy, using data that had been collected at MIT, and then applied that policy (without any modifications) to a completely different environment at CMU. The policy successfully followed natural language directions through an unknown unstructured environment, and also demonstrated backtracking to recover from errors.

CoBot has already demonstrated long-term deployments and many real world cases of user interaction, which makes it an ideal platform for investigating future integration work with our approach. Adding a control interface for CoBot driven primarily by speech and natural language would provide extensive data on real world usage and performance of our approach to following natural language directions. Furthermore, since people already rely on CoBot to perform tasks, it would serve as a great experimental platform to extend our approach to other tasks (beyond following directions), and improve human-robot interaction through natural language.

Second, our demonstration of the semantic mapping capabilities from Chapter 5, combined with our belief space policy from Chapter 6 on two different robots (the Husky and the MIT autonomous wheelchair) showed that inferring a distribution of maps (from the information contained in the natural language) and using it to follow the directions can lead to performance that is comparable to following directions with a completely known map. These platforms use a camera-based perception system and a laser-based online mapping framework, with the help of fiducials for object classification. We demonstrated the ability to reason about objects (e.g., cones, hydrants), regions (e.g., hallways, kitchens), and the relationship between them (e.g., behind, down). Our approach is agnostic to the specific perception and low-level planning systems on the robot platform.

We presented physical experiments on the Husky and Wheelchair across several different scenarios, repeating multiple trials on both platforms. Additionally, our simulated experiments provide more extensive results, and more importantly show that our approach performs well across a wide range of parameters such as the sensing range. These results demonstrate that language contains useful information about the world that can be used to infer a distribution of maps. Additionally, we showed that this approach can handle ambiguous directions, such as a direction that does not specify *which* object the goal may be behind. By updating the distribution of maps and reasoning in belief space as the robot gathers more information, our approach can recover from false initial assumptions (in this case, going behind the wrong object) to successfully follow complex natural language directions.

In the following chapter we present another real world application of our approach to understanding natural language instructions, this time in a large-scale outdoor environment. This system integrates a complete mapping and perception pipeline (with semantic image labeling and 3D LIDAR scanning), a cognitive architecture for prediction and long-term planning, and our natural language understanding approach. As we will show, this complete integrated approach enables us to successfully follow complex natural language directions through unknown unstructured outdoor environments.

Chapter 8

Integrated Demonstrations on an Autonomous Outdoor Robot

> To achieve great things, two things are needed: a plan and not quite enough time.
>
> Leonard Bernstein

We now present an extension of our work on imitation learning for natural language understanding to a robot operating in a large unstructured outdoor environment. As in previous chapters, the robot starts with no knowledge of its environments and must make decisions using the information it has collected so far. The perception systems in this chapter are a combination of an image-based semantic classifier and a 3D point cloud object detector. One key difference from our previous approach is that in this chapter we learn to ground language to the space of *plans* instead of mapping language to actions:

$$\pi = \underset{\text{plans}}{\operatorname{argmin}}\ \operatorname{cost}\left(\text{plan} \mid \text{language}, \text{world}\right). \tag{8.1}$$

The world representation in this chapter is a more complex representation of the environment the robot is operating in. It will include objects detected by the various perception systems running on the robot, as well as predictions of some parts of the environment. After the robot selects the minimum cost plan, it begins to execute it.

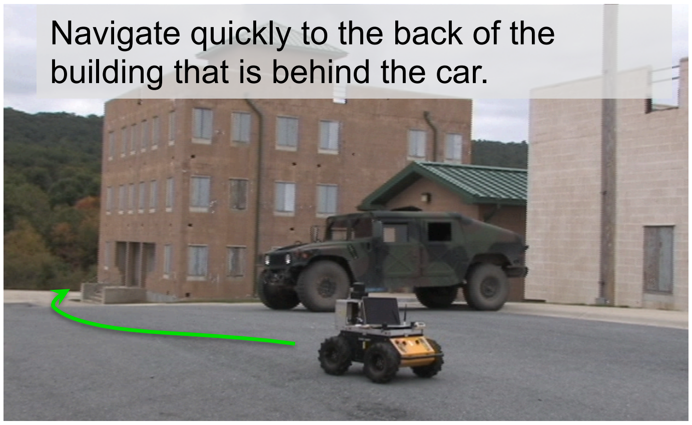

Navigate quickly to the back of the building that is behind the car.

Figure 8.1: The Husky robot platform executing a natural language command in an outdoor environment.

As in prior chapters, we will continuously update the world model and replan as the robot gathers more information about the world.

This chapter is drawn from a previous publication by Oh et al. [111] as part of the Robotics Collaborative Technology Alliance (RCTA) project. We present a summary of our approach and the system description as a service to the reader, and defer to the paper for full details. The specific contributions of this dissertation to the RCTA project are:

- a cost-based planning algorithm that takes into account language landmark constraints (e.g., “left of,” “around”), or a navigation mode (e.g., “quickly,” “covertly”),
- a method for learning the planner cost function using imitation learning, and
- features for reasoning about the navigation mode.

We begin with an overview of all the system components in Section 8.1, then present experimental results collected over several field trials in an outdoor training facility consisting of a simulated town in Section 8.2. We then highlight key differences to our previous approaches and present a summary in Section 8.3.

This chapter addresses natural language command of an autonomous mobile robot operating in an outdoor environment consisting of buildings, cars, and other objects such as traffic barrels and fire hydrants. For example, we show one possible command in Figure 8.1. In addition to specifying a goal landmark, the natural

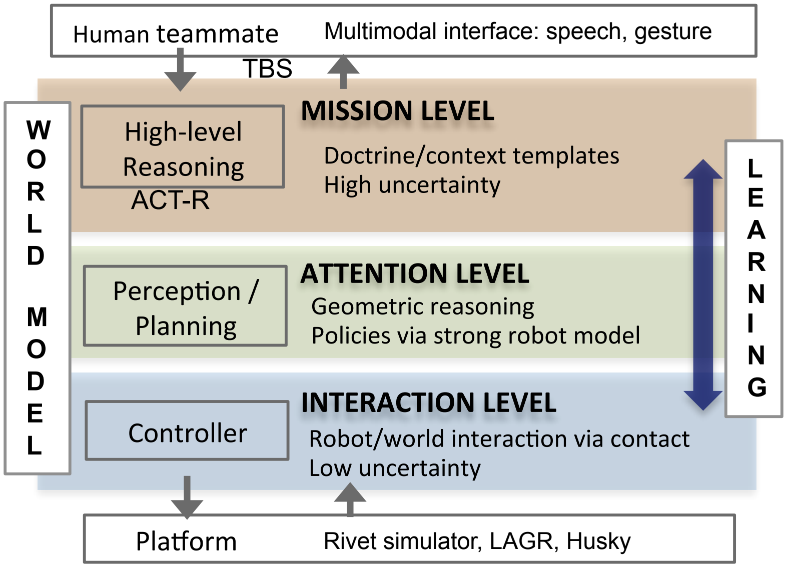


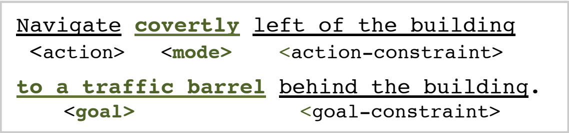
Figure 8.2: Our intelligence architecture for human-robot teams.

language command also describes any specific constraints to obey during execution (e.g., “keep left of the car”), and also a desired behavior for the robot to maintain while it is traveling (e.g., moving “quickly” or “covertly”). The robot must then infer a plan that takes it to the intended destination, while obeying the explicit constraints and exhibiting the desired behavior.

8.1 System Overview

We have integrated multidisciplinary components into a combined intelligence architecture that enables a robot to perform high-level cognitive tasks specified in natural language in an unknown environment. We leverage recent developments from the fields of cognitive architectures, semantic perception, and natural language understanding. This architecture is platform-independent and flexible to implement various tactical behaviors (for example navigation, search, or manipulation).

Our system architecture, shown in Figure 8.2, is tightly coupled to the world model [37]. The architecture consists of a hierarchy of tasks at three levels: *mission*, *attention*, and *interaction*. Every task is a computational node that encapsulates a particular functionality: each node interleaves perception, planning, and execution



Navigate covertly left of the building to a traffic barrel behind the building.

Figure 8.3: An annotated Tactical Behavior Specification (TBS) command. The TBS is a semantically annotated clause, similar to Spatial Description Clauses.

monitoring. At each level, the world model stores all of the data for matching the task's pre- and post-conditions. The world model also stores resource models for the robot, the current task/subtask execution trace (for monitoring and inspection), and the history of this trace (for offline learning).

Mission Level: The tasks at the mission level implement specific doctrines, and mimic human functionality. The tasks determine how to sequence atomic actions at the next level down in order to achieve the mission. The basic actions are predetermined, but the challenge is to figure out when and how to apply these templates to particular cases, or to modify actions based on situational awareness and context. The world model includes state data that represents contexts and situations as well as tasks and subtasks that are planned and being executed.

Attention Level: The mission-level tasks call attention-level tasks to navigate from place to place, grasp and manipulate objects, and perceive semantic objects and hazards. Rather than encoding doctrines, the reasoning at this level is primarily geometric. For example, a task reasons about how to move a manipulator to avoid obstructions and get into position to grasp an object. The world model includes grids of hazard data (interpreted relative to a particular robot model), semantic objects such as doors or buildings, scrolling maps, and planned/executing tasks and their sub-goals.

Interaction Level: The attention-level tasks call interaction-level tasks to move the robot. Tasks at this level are essentially controllers, and typically cycle at 10 Hz or faster. At this level, the world model includes robot kinematics and dynamics, as well as metric data such as geometric shape, appearance, and material properties for objects in the world.

Natural Language Input

The system accepts as input free-form natural language commands. Just as we did in Section 3.1, we convert the natural language command into a sequence of structured clauses. For this application, we developed a new structured clause called the Tactical Behavior Specification (TBS). It consists of the following fields:

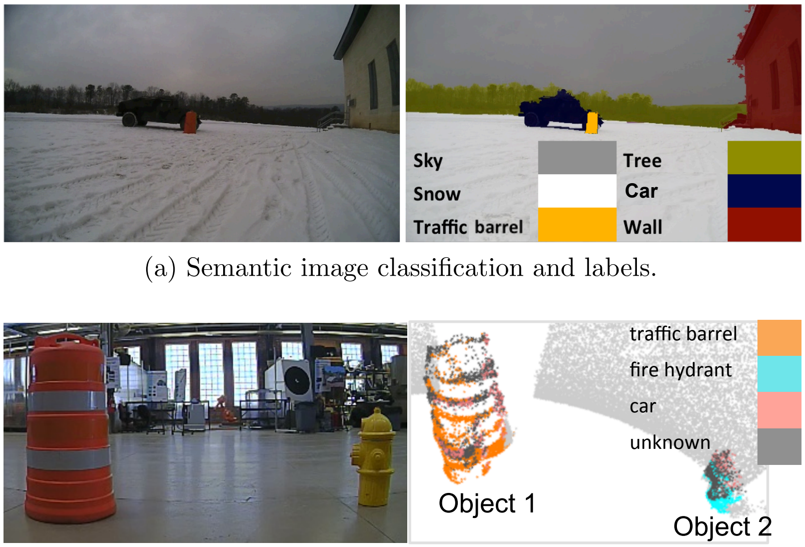
- action: the high-level behavior. We focused on navigation, but this could be a search or observe behavior
- goal: a landmark in the world that specifies the destination
- goal constraint: relative location of the goal landmark with respect to other landmarks in the world (to clarify which landmark to use as the goal)
- action constraints: list of constraints to obey while following the direction
- mode: desired behavior to obey while performing the task

By extracting these semantically meaningful terms from the command, we will again be able to reason separately about the various components in the command. For illustration, Figure 8.3 shows the TBS representation for the natural language command "navigate covertly to a traffic barrel behind the building while staying left of the building." In this work we convert the natural language direction into this TBS form using a variant of the Hierarchical Distributed Correspondence Graph (HDCG) model [59].

Perception

The robot (Clearpath$^{\mathrm{TM}}$ Husky) is equipped with a 2D camera (Adonis) and a 3D scanning LADAR sensor (General Dynamics XR). The LADAR sensor is mounted 0.7 m above ground which creates approximately 4 m radius dead zone around the robot. To remedy this, we installed a 2D scanning laser (Hokuyo UTM-30LX) at 0.25 m for obstacle detection in this dead zone. The camera is used for scene classification and identification of objects in the robot's field of view. The perception labels are: wall, grass, asphalt, concrete, traffic barrel, car, gas pump, and fire hydrant. The output of the classifier is then mapped to the LADAR point cloud to render a colorized point cloud.

Our camera-based scene classifier was developed by Munoz et al. [107]. This



(a) Semantic image classification and labels.

(b) Point cloud object detection.

Figure 8.4: Semantic perception capabilities on the Husky robot, using a 2D image classifier (8.4a) and a point cloud based classifier (8.4b).

method operates by super-segmenting the image, predicting a distribution of labels for each small region, and refining the classifier at progressively coarser levels. We use the most probable label for each of region to label the image, Figure 8.4a shows one classified image. Since this scene classifier labels pixels but does not detect individual objects, we project the image classification into the 3D point cloud to separate labeled pixels into discrete objects.

The 3D perception has two purposes: detecting walls and clustering pixels into objects. To detect walls the system looks for planar surfaces using RANSAC [48]. Neighboring walls can either be merged or connected via corners. This will provide the basic structure that compose buildings. To form objects, we cluster pixels using the Euclidean distance in the Point Cloud Library [136], and use Naive Bayes to classify each cluster according to the distribution of class labels generated by the image classifier, as shown in Figure 8.4b.

Prediction

Because the robot is operating in an unknown environment, it will not have complete knowledge of the landmarks around it. Our grid-based planning algorithm requires

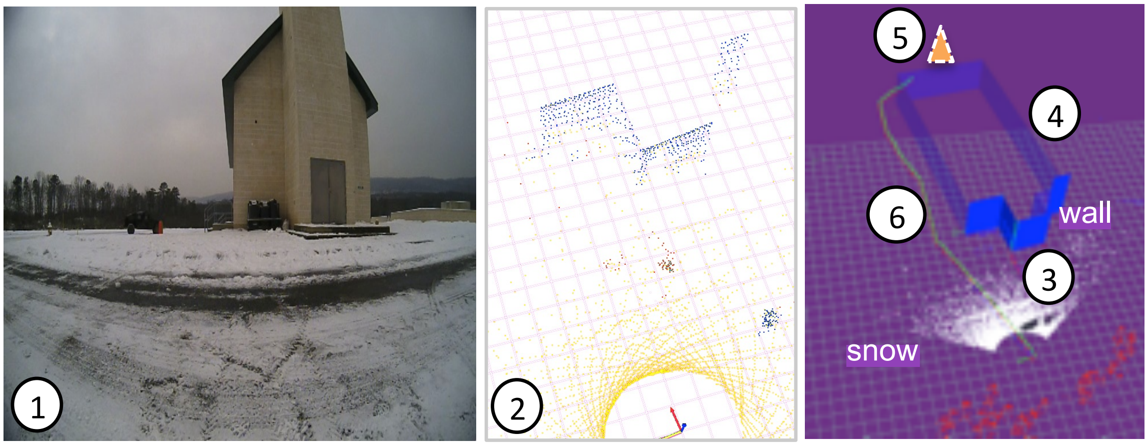


Figure 8.5: Building and landmark prediction using partial information. Using a view of the front of the building (left), the robot extracts from its point cloud clusters of walls (center). Then, the cognitive architecture predicts (right) the remainder of the building based on examples in its declarative memory, and the location of the goal landmark (traffic barrel) using the information in the TBS.

a complete world representation, so we predict unknown parts using the sensor information available to the robot. The prediction module attempts to predict two things: the complete shape of buildings around the robot (even though it may only see part of it), and the possible locations of unknown landmarks that would obey the natural language command.

The robot first predicts the entire shape of the building using clusters of walls it has previously detected (described above). For example, in Figure 8.5 the robot can initially only see the front of the building. To reason effectively about where the goal could be, it generates a prediction of the entire shape of the building, including the parts it cannot yet see (in this case, the side and back). We use a cognitive architecture (ACT-R [7]) to generate this prediction, by matching the data it has to geometric patterns in its declarative memory. These patterns are compiled from training data of buildings around the test site. We update the predicted shape of the building as it gathers more information.

Once we have a complete prediction of the building geometry, we predict the possible location of any necessary unknown landmarks. More specifically, if the goal landmark has not been detected we use the information contained in the TBS (the goal and relation fields) to determine a possible location for the goal. This predicted object will be used by the assumptive planner to plan a path to the goal.

Natural Language Reasoning

Once the perception system detects objects and the prediction module generates a (possible) complete model of the environment, our natural language reasoning component first grounds the landmarks in the TBS to objects in the world, then produces a costmap for the TBS constraints. The planner will then combine this costmap with others (such as obstacle avoidance) and optimize the combined costmap to generate a full path to the goal.

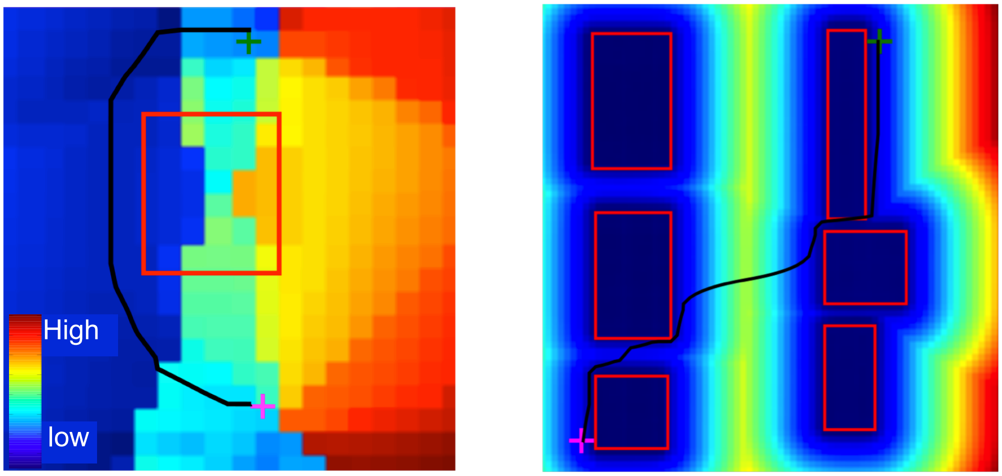
The grounding component addresses the question of *which* landmarks in the world correspond to the ones in the TBS. We associate each landmark in the TBS command with an object in the world model by taking into account the landmark name (in the TBS) and the object label (its semantic class). We additionally use any given spatial constraints (the goal constraint TBS field) to reason about the relationship between possible goal objects and neighboring objects, using spatial features describing the relative geometry between both objects (e.g., “behind the car”). This component grounds the landmarks described in the TBS to objects in the world model.

The spatial navigation components addresses *how* to travel such that the resulting path obeys the linguistic command. We treat this problem as learning a mapping from linguistic terms (such as “left of,” “around,” or “covertly”) to a cost function f that can be used to generate the costmap for any cell in the grid. More specifically, given a TBS spatial term σ, the robot solves the planning problem of finding the minimum cost path ξ^* under the cost function f_σ:

$$\begin{aligned} \xi^* &= \operatorname*{argmin}_{\xi \in \Xi} f_\sigma(\xi) && (8.2) \\ &= \operatorname*{argmin}_{\xi \in \Xi} w_\sigma^T \phi(\xi) && (8.3) \end{aligned}$$

where the set of valid paths is Ξ, and we assume that the cost function f_σ takes the form of a linear sum of features ϕ under weights w_σ. The features describe the shape of the path, the geometry of the landmark, and the relationship between the two [158].

We use imitation learning to learn the weights w_σ from a set of N demonstrated



(a) Constraint "left of."

(b) Mode "covertly."

Figure 8.6: Learned navigation cost functions and resulting paths (drawn in black, starting at the pink plus symbol) for the navigation spatial constraint "left of" and the navigation mode "covertly". The red rectangles are outlines of buildings.

paths $\{\hat{\xi}_i\}_1^N$. To do so, we minimize the difference between the cost of the expert's demonstrated path $\hat{\xi}$ and the minimum cost path under the current cost function. This can be written as the following loss function:

$$\ell\left(w_\sigma, \hat{\xi}\right) = w_\sigma^T \phi(\hat{\xi}) - \min_{\xi \in \Xi} w_\sigma^T \phi\left(\xi\right) + \frac{\lambda}{2} \|w_\sigma\|^2 \tag{8.4}$$

with regularization parameter λ. This Max Margin Planning formulation [121] learns the cost function that we use to generate paths obeying the natural language constraints. Figure Figure 8.6 shows two sample costmaps for the terms "left of" and "covertly." Note that the covert mode is not a well-specified concept, but we can nevertheless learn this behavior from demonstrations of desired paths. The resulting costmap for each behavior can be used by the planner to generate a minimum cost path (shown in black) that reasons about the user's natural language constraints.

Table 8.1: Several TBS commands used in the experiments. These commands are in semi-structured English.

Navigate quickly to a car that is near the fire hydrant.
Navigate quickly to the building that is near the traffic barrel.
Stay to the left of the building; navigate quickly to a traffic barrel that is to the back of the building.
Stay to the right of the car; navigate quickly to a traffic barrel that is behind the car.
Stay to the left of the building; navigate quickly to a fire hydrant that is to the left of the building
Stay to the right of the car; navigate covertly to the right of the building that is behind the car.

8.2 Experimental Results

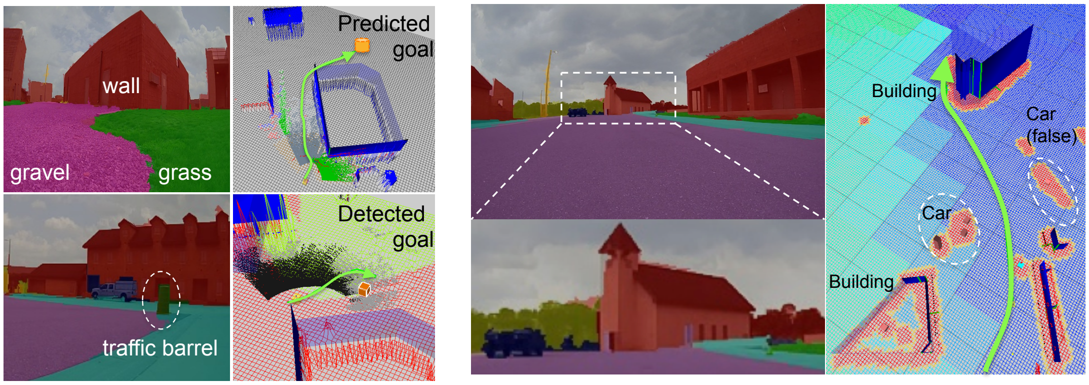
The examples and results reported in this section are based on outdoor experiments conducted in a 1 km^2 training facility located in central Pennsylvania, that includes 12 buildings in a simulated town. The simulated town contains a gas station (with gas pumps), and we can further vary the situation by moving cars, traffic barrels, and fake fire hydrants into the scene.

The dates of the experiments spread between December 2013 to August 2014 with varying conditions in terms of weather, sunlight, background, and terrain conditions. This included several runs after heavy snow, arguably more uncomfortable for the experimenters than the Husky robot. In these experiments we used semi-structured English commands as commands to the robot, Table 8.1 lists some of the commands used. We commanded the robot with 30 unique natural language commands over 46 different runs. We present a summary of the experimental results in Table 8.2. A majority of the runs were successful, and traveled a significant distance through the initially unknown environment. Common causes of failures in the 11 incomplete runs included network communication problems, platform errors, or operator intervention (due to unsafe commanded plans).

To illustrate two individual runs, we show in Figure 8.7 visualizations from different

Table 8.2: Overview of results on the Husky robot platform.

Number of unique TBS commands	30
Total number of runs	46
Number of successful runs	35
Number of incomplete runs	11
Distance traveled per run (m)	21.0 ± 14.3
Number of runs traveled more than 30m	11



(a) Navigating to a traffic barrel.

(b) Navigating to a building.

Figure 8.7: Illustration of two separate runs. The commands are "Keep to the left of the building; navigate to a traffic barrel that is behind the building" (8.7a), and "Keep to the right of a car; navigate to a building that is behind the car" (8.7b).

parts of the run for two commands. The first command ("Keep to the left of the building; navigate to a traffic barrel that is behind the building" in Figure 8.7a) shows the robot detecting the front two walls of the building (top left panel) and predicting the rest of the building using the cognitive architecture (top right). In addition, the robot predicts a goal location and then plans a path to this goal that obeys the constraints ("left of the building") specified by the command. After the robot rounds the corner it detects the actual traffic barrel (bottom left panel). It replaces the predicted goal with the detected object and executes the remainder of the plan (bottom right).

The second command ("Keep to the right of a car; navigate to a building that is behind the car" in Figure 8.7b) shows one of the longest runs the robot completed. The robot first had to detect cars in order to determine which building to navigate to. Even though the robot incorrectly perceived a car on the right, it correctly associated the TBS action constraint landmark to the correct object in its world model. The right panels shows the resulting coarse costmap for the constraint ("right of the car") and the final path.

8.3 Chapter Summary

This chapter presented an extension of our work on natural language understanding to large-scale outdoor environments. This dissertation's contributions were a small part in a much larger architecture that includes a full semantic perception system, a cognitive architecture for prediction and long-term planning, and spatial reasoning. Each individual module interfaces with a common world model. This chapter presented several anecdotal runs of the robot in a variety of real world conditions.

This project taught us some valuable lessons. First, it was very important that our approach to natural language understanding make few assumptions about the structure of environment or robot capabilities. Our general approach was easy to extend to this scenario, even though the optimization problem was significantly different (optimizing over paths, not single-step actions). Second, the scenarios presented here highlight the need for prediction of the relevant parts of the world (unknown parts of the building and goal landmarks). Third, the tight integration necessary between all the components required many iterations of implementation and testing in the field.

The work in this chapter is another step towards enabling human-robot teams to work together towards increasingly complex tasks. To enable robots to become full-fledged teammates, they will need to perform complex tasks that requires a semantic understanding of the world, high-level cognitive reasoning, natural language understanding, and the ability to work in unknown environments.

Chapter 9

Summary, Contributions, and Future Work

There are other Annapurnas
in the life of men.

Maurice Herzog

The experimental results presented in Chapters 7 and 8 provide experimental validation of our framework for following natural language directions across a variety of robotic platforms. These robots operate in a variety of environments (some of which are novel to the robot) and follow a variety of natural language commands. We demonstrated that robots can understand directions in completely unknown environments, discovering landmarks as they follow the direction. If the command contains implicit information about the environment (for example, the location of a landmark the robot cannot yet see), the robot can use this information to infer a suitable distribution of world models. Additionally, the corpora-based experiments from Chapters 4 and 6 demonstrate our approach in a more controlled (and repeatable) setting, with many more natural language directions.

We finish by summarizing this dissertation (Section 9.1) and its contributions to the field of robotics (Section 9.2). We then present several interesting directions for future research (Section 9.3), and conclude with some parting thoughts (Section 9.4).

9.1 Dissertation Summary

This dissertation addressed the problem of enabling autonomous robots with realistic perception to understand natural language directions. The prior approaches we studied (Chapter 2) fall into two broad categories: approaches that require a full map of the environment ahead of time, and approaches that do not require a map but have only been applied to highly structured environments (such as simulated environments or buildings without landmarks). Following natural language directions in unstructured unknown environments is a difficult problem because the robot must make decisions using a partial representation of the environment that it builds up incrementally. The robot must take a sequence of actions that explore the environment and follow the natural language direction.

We formulated this problem as one of decision making under uncertainty (Chapter 3), in which a policy commands actions to the robot using its knowledge of the world so far. We first made the problem of understanding natural language tractable by exploiting the structure inherent in directions: they are sequential and each clause refers to different meaningful terms. In this dissertation, we exploited this structure by modeling a complex natural language direction as a sequence of semantically annotated clauses (Spatial Description Clauses). Each clause contains an annotation of the verb, landmark, and spatial relationship present in the direction. In our later work, we extended this structure to include information about navigation landmarks the robot would see along the way. This decomposition enables the policy to reason efficiently about long directions.

The policy relies on a partial map of the environment that combines metric, topological, and semantic layers. This map is then used to generate a set of valid actions for the robot. Some actions represent information gathering (traveling to an unknown place), backtracking (traveling to a previously visited place), or declaring the robot has finished following the direction (the stop action). Explicitly reasoning about the stop action enables us to follow directions in unknown environments without exploring the entire environment. The policy selects the lowest-cost action to execute on the robot, using a cost function that is a linear combination of features of the state-action pair. These features numerically relate the natural language command to the action under consideration, and more importantly features enable the policy to generalize to new directions and even new environments.

We presented an approach for training the policy through imitation learning, using human demonstrations of following directions (Chapter 4). Imitation learning is especially well suited for unstructured environments: engineering (and tuning) a system to reason about a complex environment would be time consuming, whereas collecting human demonstrations is much less expensive. We demonstrated it is possible to use imitation learning to train policies that follow natural language directions, and that the resulting policy encapsulates a person's complex decision making when following directions.

The training phase of our approach collects the policy's decisions in an unknown environment. While this approach requires more implementation effort, learning in unknown environments trains the policy to reason about the uncertainty in the map, and ensures the data available to the policy at test time is similar to what the policy was trained on (as opposed to approaches that train on complete environments and remove the map at test time).

Furthermore, the policy can learn to recover from mistakes by using demonstrations of both successes and failures. This ability to recover is especially important when dealing with unknown environments, as the policy is more likely to make a mistake. Learning this requires making mistakes at training time and receiving demonstrations of the correct behavior. While the expert demonstrations generally do not contain failures, we showed a simple approach for inferring a complete expert policy and use it to include training examples from states where the learned policy made a mistake. Using training data from both good and bad states enables us to learn a policy that can recover from errors.

We evaluated our approach on a corpus of directions in Section 4.5, where we demonstrated the performance of policies that follow natural language directions through unstructured unknown environments. Through quantitative and qualitative results, we showed the benefit of various features in our representation, as well as the importance of learning error recovery.

This dissertation then introduced the notion of language as a sensor in Chapter 5. This extends our approach to handle complex instructions that convey information about both the task (what the robot should do) but also about the world (what objects are still undetected). We proposed to use language to infer a distribution of maps that provide a hypothesized view of the environment beyond the robot's sensor range.

We showed how to use this distribution of possible environments, extending our policy to reason about a distribution of possible landmarks in Chapter 6. We presented a novel imitation learning approach for learning these belief space policies, once again using demonstrations of people following directions. We showed that this approach outperforms the standard policy when the directions contain additional information. In essence, our results show that leveraging the additional information contained in language to hypothesize a distribution of maps enables performance that is close to that of having a fully-known map.

Chapters 7 and 8 presented some illustrative demonstrations on various integrated robot platforms, operating indoors and outdoors. These highlight the generalizability of our approach across various robot platforms with different sensing configurations. We also presented more extensive simulated results that varied parameters such as the sensing range, providing a clearer picture of the performance of our approach compared to operating in a completely-known map.

9.2 Contributions

This thesis contributes a framework for understanding and following natural language through unstructured unknown environments. While prior work focused on the problems of following directions through known unstructured environments or unknown structured environment, our approach is the first to address the problem of following natural language directions through unstructured and unknown environments. In this section we highlight our scientific contributions to the fields of human-robot interaction (HRI) and cognitive robotics, grounded language acquisition, and belief space reasoning and learning. Specifically, we show how robots can follow natural language directions, use language as a sensor, as well as reason (and learn) in belief space. Additionally, this chapter will highlight several key lessons learned while conducting this research.

Following natural language directions

To enable robots to follow natural language directions through unstructured unknown environments, we presented a novel formulation of this problem as sequential decision making under uncertainty. In this formulation, a policy makes a sequence of decisions using only the partial map information it has collected. This is in contrast with planning approaches that optimize complete paths in the environment (requiring a full map), or parsing approaches that convert natural language to formal controllers representing the intent of the command (and have only been applied to highly-structured environments). Our approach is much more flexible, and has the following beneficial properties:

- our approach does not require an a priori map of the world,
- our approach makes no assumptions on the structure of the environment,
- our approach is agnostic to robot platform, sensor configurations, and underlying low-level planners,
- our approach can learn the meaning of arbitrary words (provided there are demonstrations of it),
- our approach learns policies that reason about the unknown parts of the environments,
- our approach learns policies that can recover from mistakes,
- our approach produces policies that generalize to different directions, and even new environments.

Together, these properties provide an efficient and novel solution to the problem of following natural language directions through unknown environments. Additionally, we presented a novel imitation learning approach to grounded language acquisition. Our approach trains policies to reason about uncertainty and recover from mistakes, using demonstrations of people giving and following directions.

We demonstrated that our approach learns policies that can successfully follow natural language directions in unknown unstructured environments. We have shown that these policies can generalize to new directions, and can even transfer to new environments the robot has never encountered before. We furthermore demonstrated this approach on several robots: CoBot (an indoor symbiotically autonomous robot) operating at CMU, an autonomous wheelchair operating at MIT, and the Husky robot operating in large-scale unstructured outdoor environments. Our approach is independent of the particular platform, its sensing capabilities, or low-level planner. Because of our decisions to remain independent from the capabilities or configuration of a single platform, we were able to easily generalize our approach across a variety of robots.

More generally, one of the important insights of this work is that human-robot interaction can be formulated as a problem of sequential decision making under uncertainty, with several key benefits. First, we can train a policy to make good decisions using only the information available so far, and make new decisions after the robot receives new information about the world. Second, rather than coding complex error recovery behavior, the policy can represent error recovery naturally by using additional actions. As we will show in the future work, this could include dialogue with a user. Third, this formulation is efficient: we do not need to plan for long horizon trajectories that would likely become invalid as soon as the robot gains new information. Fourth, imitation learning in the context of sequential decision making has been well studied (our approach is one such application), and imitation learning could be an efficient way to train complex HRI system. While this thesis focused solely on the problem of understanding natural language directions, this general approach of modeling human-robot interaction as making a sequence of decisions could be useful for a wide range of HRI problems.

Using Language as a Sensor

We are the first to introduce the notion of language as a sensor, enabling robots to infer a suitable explicit world model that can improve its performance.[1] We showed that some natural language commands contain both explicit information about the task (what the robot should do), but also implicit information about the world (what the robot will see). For the problem of following natural language directions, treating language as a sensor enables a robot to explicitly model the uncertainty in the unknown parts of the environment by utilizing the information already provided in the natural language direction. We then contributed a novel approach to inferring

[1] This is a joint contribution with Sachithra Hemachandra, Thomas Howard, and Matthew Walter.

a distribution of maps and behaviors from a natural language command, by utilizing the information in the language about the world to hypothesize the location of landmarks. Then, we utilize the information in the language about the task to infer a distribution of behavior constraints under this distribution of maps. We use this inferred distributions in a belief space policy that reasons about a distribution of landmarks to follow the direction. Similarly to our early work, the robot updates its world representation as it travels through the environment and senses new landmarks. Our approach of learning to mine language for additional information about the world enables performance on robots that is comparable to the performance when a fully-known map is available.

This area of research has potential for many more contributions in the field of grounded language acquisition, we will highlight some of these in Section 9.3.

Belief Space Reasoning and Learning

Our approach of leveraging the implicit information to infer a map distribution enables robots to successfully follow directions that are more complex by explicitly representing the uncertainty about the world. To handle this uncertainty, we presented a belief space policy that is an efficient solution to the problem of reasoning and acting in a state distribution. This policy is fast and efficiently makes use of the information provided in the map distribution by reasoning about the uncertainty inherent in the information available. To learn this belief space policy, we present a novel formulation of inverse reinforcement learning in belief space that uses a kernel embedding of the action feature distribution. This approach learns a belief space policy from human demonstrations, and captures the expert's behavior given the uncertainty in the environment. It enables the policy to handle more complex commands that may have some ambiguity (at least from the perspective of the robot). Our experiments demonstrated that the ability to reason and learn in this belief space enables robots to recover from false assumptions they make when following directions.

Applying imitation learning techniques to learn from a distribution of data is a similarly promising avenue for future research.

9.3 Future Work

While the contributions outlined above further the state of the art in natural language understanding for robots, there are scenarios that were beyond the scope of this dissertation that would be interesting future problems to address. Additionally, our findings raise new scientific questions that are promising avenues for future research. These future research topics lie along several dimensions: handling more varied natural language commands, dealing with more complex integrated robot systems (that may have more sources of uncertainty), and improving the human-robot interaction. In this section we detail several future work ideas, and how we might begin to address them.

Handling complex language directions

One shortcoming of this dissertation is that all commanded directions were fairly simple by design. Dealing with more complex language can be done at several levels, and a successful solution will likely be a combination of:

- mapping complex language to a simpler (SDC-like) representation. In many cases complex language contains utterances that are not necessary to follow the direction correctly.
- learning the meaning of more verbs and spatial relationships by increasing the amount of training data.
- improving the object co-occurrence reasoning component, to better handle references to arbitrary landmarks.
- building better priors over landmark locations given a language command, to better guide the policy. This could even be extended such that a robot may not need to perceive a particular landmark if the belief distribution is well localized.

These improvements would require a mix of expertise in Natural Language Processing, Semantic Mapping, Imitation Learning, and Belief Space Reasoning. Fortunately because our approach is modular (the components are independent from one another), we can replace one module easily to better handle certain cases and re-evaluate our approach.

Dealing with incorrect or highly-ambiguous directions

In this work we have assumed the directions given by the user are correct: they do not contain mistakes, and the direction contains enough information to follow it correctly. When these assumptions do not hold, our current approach will generally do poorly. The major reason is that the transition between clauses (with the stop action) is final: when the policy transitions from one SDC to the next the policy never considers it could have made a wrong decision. This wrong decision could have been caused by a single SDC that was very ambiguous (with multiple possible options), a landmark mentioned by the user that cannot be detected by the robot, or the policy could have simply made a mistake.

In this case, after it transitions to the next SDC the policy will be forced to follow the remaining clauses even if it receives more information that makes it clear the policy made a mistake early. This is because we do not currently have a way to recover from errors in the stop action. For instance, a robot following a long direction may turn into a hallway, transition to the second SDC in the direction, and after a few more actions notice the hallway is a dead end. The best response would be to backtrack *across SDCs* (back to the previous SDC), return to the original starting point, and use this newly-gathered information to better follow the direction. Enabling the policy to reason beyond myopic single-SDC segments when following directions would make this approach much more robust.

One way of accomplishing this would be to enable the policy to consider any of the previous SDCs when making a decision by reasoning in the belief space of ways to follow the direction. In this setting, the policy could consider a set of actions for all SDCs instead of only the current one, and represent the likelihood of the command being correctly-followed under the map known so far. In the above scenario, after the policy realizes the hallway is a dead end it could still consider an alternative action for the first SDC. This process could be made efficient using samples from the belief, for example using a particle filter. Essentially, this would allow the policy to consider that it made a mistake at the transition point and consider many more possibilities.

Reasoning about directions that are incorrect (i.e., contain mistakes) is more challenging. For example, people sometimes confuse directions (left instead of right), provide references to landmarks that are no longer present (e.g., “turn left where the statue used to be"), or under-specify the command (e.g., "find the stairs"). We can already begin addressing some of these issues using our map inference approach, especially for cases relating to landmarks (by hypothesizing possible locations and following the direction in a distribution of maps). For completely *wrong* directions, one approach would be to learn a model of how people make mistakes when giving directions, and combine it with the ability to reason across the entire direction described above. When the robot begins to observe an environment that does not match the given direction, the possibility that the direction contains a mistake would increase.

Grounding to task parameters

In Chapter 8 we mapped language to the navigation mode for a robot navigating to a goal (in this instance, quickly or covertly). These language-based task parameters do not explicitly prescribe what to do, but rather *how* to do it. There may be interesting scenarios where the task is known, but we can use language to determine the parameters of the task. Manipulation is another setting where language can be used to convey information about *how* to grasp or manipulate an object (e.g., gently, keeping the object upright, staying away from something, etc.). While some of these task parameters are evident, others may require learning from demonstrations.

Increasing the amount of natural language interaction

While this dissertation is an important step towards effective human-robot collaboration using natural language, a strong limitation is that the natural language interaction only happens once, and always from the user to the robot. The user commands the robot with a direction, and the robot is expected to execute it without any further information from the user. Two ways of increasing the amount of natural language interaction are enabling the robot to take further directions from the user during execution, and adding a dialogue component.

When the user is monitoring the robot during execution (for example while sitting on the autonomous wheelchair), he or she has the opportunity to provide more detailed descriptions of the environment, or even give further directions while the robot is executing the desired task. For example, the user could provide a rough direction at the beginning, and then provide more accurate descriptions of the environment when they are relevant. Additionally, the user could modify their instruction or provide a new one entirely. One such modification could occur if there is some ambiguity and the robot (unsure about where to go) selects the wrong option (for example, the robot chooses the wrong hallway to travel down). In this case, the user could easily provide a modification to the task (for example, saying "not that hallway"). Our current formulation could handle both of these additional inputs by enabling the semantic mapping framework to continuously accept and process new annotations.

Human-robot dialogue would be another avenue for increasing the amount of language driven interaction. Many approaches have studied dialogue in human-robot teams, and ours is well suited to the problem because we represent the uncertainty in the world explicitly (by reasoning about the distribution of possible environments). When there is high uncertainty in this representation and the robot could gain a lot of information from the user, asking a question may be one possible action the robot considers. This would enable the robot to ask clarifying questions or request further information from the user.

Applications to other tasks and domains

This thesis focused on the problem of following natural language directions through unknown environments. However, the techniques of using language to command robots and learning a sequential decision making policy could be applied to other tasks and domains. For example, cooperative manipulation tasks (where the robot is assisting a user by carrying something) would benefit from a natural language interface. The user can explicitly describe what they expect the robot to do using a natural language command, without relying on complicated operator interfaces or implicit information like forces.

Similarly, natural language could enable robots to perform complex household tasks in assistive settings. These robots may need to perform a variety of tasks ranging from manipulation (e.g., picking up an object), navigation (e.g., autonomous wheelchair going somewhere), search (e.g., looking for an object), and many others. Rehabilitation robotics will be an increasingly fruitful area of research into effective interaction modalities, and it is likely that speech will play an important role.

Reasoning about perception uncertainty

We purposefully limited the scope of this dissertation to not include reasoning about perception uncertainty, but this area holds the promise for more future work. In this thesis we used several perception systems: a simulated perception system (in Section 7.1), a fiducial-based perception system (in Sections 7.2 and 7.3), and a fully-integrated semantic perception system (in Chapter 8). However, our model of the world assumes that the robot's perception system only returns the object's landmark and geometry, with no uncertainty (these can update as the robot moves, but we do not explicitly make use of any uncertainty information from the perception system).

Dealing with perception uncertainty is a challenging problem and will require improvements to the policy. One approach would be to optimize over the expected perceived object, by taking into account the likelihood of various possible observations. Alternatively, using knowledge about the relationship between nearby objects (the context) would provide a possible solution to reasoning about ambiguous perception data. In this setting, the policy could use its knowledge about which objects occur near each other in indoor environments to filter out unlikely detected objects that may be due to an uncertain perception system. The natural language command can also provide information about what the robot is perceiving (continuing the theme of using language as a sensor).

Another approach is to utilize more accurate natural language descriptions of the world: if the user can describe in enough detail where the landmark is, it may not be necessary to ever sense it. This may require some dialogue between the user and robot, to localize (roughly) the location of the landmark. One interesting setting to consider would be a "blind robot" navigating using its belief about the location of navigation landmarks, without ever detecting their actual location. Under our belief space planning formulation, the robot should be able to follow directions using objects it will never sense, using only a distribution over its possible position.

A third possible improvement would be in the area of active perception. Currently any observation is opportunistic as the robot moves through the environment. Planning for perception could improve this, so that the policy actively seeks out areas of high information gain. This would require representing in the policy the robot's perception capabilities (e.g., field of view), and train the policy to sense by reasoning about this information.

Reasoning about other sources of uncertainty

One source of uncertainty we do not yet handle is in the *task progress*, due to incomplete knowledge of the world. When following directions, the destination (or a landmark in the next clause) could appear before we expect it. This should cause the robot to transition to the appropriate place in the direction's sequence, without necessarily following all previous SDCs. In more complex tasks, unexpected events may similarly cause our estimate of the task progress to be incorrect (for example, finishing the task early because of external events). These situations are good examples where a policy is beneficial because it uses the latest available information to make the next decision. In these scenarios, explicitly reasoning about the uncertainty in the task's progress could handle this type of incomplete information and enable the robot to detect when the task is complete.

Our solution of using a policy is not restricted to operating in unknown environments: even when full world knowledge is available, there may be other aspects of the problem that are uncertain and warrant a sequential approach. For example, in settings where actions have consequences there is uncertainty in *action* transitions. Manipulation is one such domain, where an action could succeed or fail in unexpected ways (resulting in an uncertainty in the world state). Similarly, external agents in the world may introduce uncertainty into the *state* transitions, for example through their actions in a collaborative assembly task. Planning single decisions before re-evaluating the state of the world may be a useful approach to take in these types of scenarios where reasoning about the uncertainty directly is difficult.

Gathering information about the world through user interactions

We have shown that we can leverage the user's natural language command to infer information about the environment (i.e., learning a map distribution from the implicit information in the command). More generally, we have shown that learning about the world through user interactions improves performance. Other approaches have learned environmental knowledge from people, using dialogue [75] or natural language descriptions [168]. Extending this idea further, there may be other aspects of the world we can learn about from user interaction (whether direct or indirect). For example, observing where people go (and what they do) in an environment could provide information about the semantic properties of places (e.g., learning where the kitchen is). This opportunistic information gathering through user interactions would improve a robot's reasoning capabilities.

Another source of user information robots could mine is information that is available on the web. For example, several descriptive directions to different offices in a building could be used together to generate a rough map of the building layout. Similarly, non-annotated maps (e.g., floor plans and crude hand-drawn maps) could be paired with natural language instructions to provide a better view of the environment.

Learning feature representations

The feature representation we used in this work was engineered. It may be possible to instead learn the feature representation the policy uses to compute the cost of each action from training data. Ideally this learning feature representation would encapsulate enough information to represent the necessary information to map from action to costs, without requiring hand-tuning or feature engineering effort. Deep learning is a promising avenue that has been used for learning the feature representation to understand the content of images [79], speech recognition [58], and even perform automatic image captioning [68, 166].

Additionally, it is possible that the policy could use high-level semantic information to determine which features are most useful and should be computed. For instance, features that determine whether or not a path intersects with a landmark may not be useful for directions that do not have the word "through." In these cases, only computing the relevant features may reduce the amount of "noisy" features the policy must current learn to ignore. This may improve the performance of the system when following directions by only "turning on" features that are meaningful for the given command.

Teaching tasks using language

In this dissertation we trained robots to complete a task expressed in natural language, using demonstrations of the correct behavior for the task. While in this work we demonstrated several strategies for providing many training examples from a small amount of expert interaction (namely, only using a single path), collecting these demonstrations is still time consuming. One possible improvement would be to teach the task *solely* using language. In this setting, the robot would perform some actions, and receive feedback in natural language. This feedback might be a simple indication of how well the robot is performing (green light / red light), or modifications to the current behavior expressed in natural language (e.g., "that was a left turn," or "those are the elevators, not the stairs"). This approach would build on work in grounded natural language acquisition [28, 78, 103] where a robot attempts to learn the meaning of words and perceptual or world attributes simultaneously, and extend it to learning tasks. While it would likely require more user interaction to properly learn a task, it could be applied to more diverse scenarios where providing demonstrations may be complicated.

9.4 Conclusions

This work is one step towards enabling real robots to understand natural language in unknown environments. Prior approaches to the problem of following directions either required a fully-known map of the world, or a highly-structured environment. These constraints severely limited the environments in which these approaches could operate. If our goal is to have robots operating in everyday non-specialized environments without a priori maps, then we need to improve their ability to reason about unstructured and unknown environments.

This dissertation addressed the specific problem of enabling autonomous robots to follow natural language directions through unknown unstructured environments. We first exploited properties of spatial language directions to model directions as a sequence of semantically annotated clauses. We then formulated direction following as sequential decision making under uncertainty, and solved it using a policy that is learned from human demonstrations. This formulation is quite general, and could be applied to many other problems in natural language understanding and human-robot interaction.

Additionally, we showed that language can be used as a sensor to enable robots to explicitly reason about the unknown parts of the environment. Robots can infer a distribution of possible maps beyond their perception range, using information from their sensors and natural language descriptions of the world. More generally, we have shown that we can learn both a *task* and suitable *world models* through user interaction (in our work, language). This ability to utilize the information contained in a command will become more important as people will expect robots to reason about complex tasks including about the parts of the environment the robots have not yet observed.

People and robots are increasingly working together towards common goals in shared environments. This rise in human-robot interaction will require new modalities for controlling robots: ones that go beyond the traditional methods of programming languages, complex interfaces, or extensive user training. Enabling robots to understand natural language instructions would allow lay users to command complex robots in diverse settings such as health care, home assistance, search and rescue, and factories. This would remove one of the major barriers to having autonomous robots interacting ubiquitously with non-expert users.

Appendix A

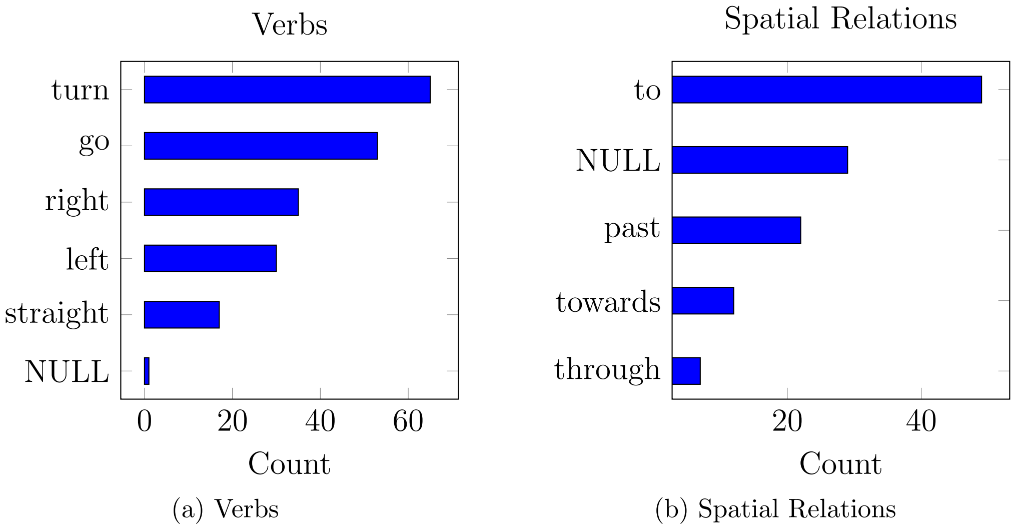
Corpora of Natural Language Directions

As we discussed in the results section of Chapters 4 and 6, we evaluated our approach on two corpora containing multi-step directions. This appendix details the contents of these corpora: the verbs, spatial relations, and objects used in the directions. We also list the entire set of directions used.

A.1 Corpus of Basic Natural Language Directions

The basic corpus of directions is used in the results presented in Section 4.5. The directions in this corpus are relatively simple, as we purposefully limited the verbs and spatial relationships allowed in the directions. Within the restricted space of language, the directions are still fairly rich: there are multiple ways of expressing the same command, objects in the world are referred using different names (e.g., “sofa” vs. “couch”), and verbs occur alone or paired with spatial relationships.

We show the relative occurrence of verbs (Figure A.1a), spatial relations (Figure A.1b), and landmarks (Figure A.3). These show the contents of all SDCs in the directions. Additionally, Figure A.2 shows statistics for the total distance each direction travels and the number of SDCs per direction across this corpus.



(a) Verbs

(b) Spatial Relations

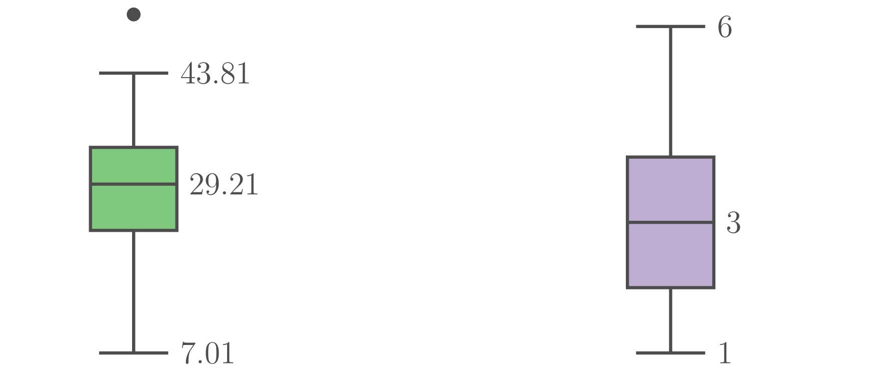
Figure A.1: Verbs and spatial relations used in the basic corpus of directions.

The landmarks are particularly interesting: many SDCs do not use a landmark (e.g., "turn right"), while some landmarks only occur once or twice across all directions. This means it is highly likely that the policy at test time will need to reason about a landmark it has never encountered before. Additionally, many landmarks mentioned in the directions are synonyms, for example photocopier and copier, or seats, couch, couches, and sofa. The environment (or perception system) will only contain a single label for all such instances, so the policy must learn to generalize across different ways of referring to the same object. The verbs in the SDCs contain many instances of the non-descriptive "go," sometimes paired with a spatial relation.

List of basic directions

Below is the complete list of 40 directions used in the basic corpus:

- turn right towards the stair, go straight towards the cabinet.
- turn left to the sofa, turn right to the window.
- go straight towards the sofa, turn right to the cabinet.
- go straight to the whiteboard, turn left, turn right to the tank.
- go through a door, turn right towards the desk, turn left, go straight past the



(a) Path length statistics (m).

(b) Number of SDCs per direction.

Figure A.2: Distribution of path lengths and number of SDCs for directions in the basic corpus. (40 directions)

whiteboard, turn left.

- go past the trash, go straight to the window.
- turn right, go past the fountain, go straight to the tank.
- go past the stair, turn left towards the mailbox, turn right, turn left to the sofa.
- turn right, turn left to the fountain, go to the sofa.
- turn left to the whiteboard, go past the fountain, go straight to the couches.
- turn left, go past the whiteboard, go to the cabinet.
- go straight towards the fridge, turn left, go to the whiteboard, turn left to the tank.
- go past the trash, turn right, go to the desk.
- turn right to the whiteboard, go straight to the fridge.
- turn right, turn left to the elevator, go to the couch.
- go past the fountain, turn right to the trash, go to the cabinets.
- go past the tank, turn right to the photocopier.
- go past the fountain, turn right to the elevator.
- go past the bathroom, turn right to the stairs.

- go past the bathroom, turn left.
- turn right to the bathroom.
- turn right to the whiteboard, go past the water fountain, go to the couch.
- turn left to the seats, go to the cabinets.
- go past the elevator, turn right, go to the copier.
- go past the trashcan, turn left, go straight to the staircase.
- turn left to the elevator.
- turn left, turn right to the mailbox.
- turn right, turn left to the table.
- turn right, turn left to the sofa, turn right to the window.
- go straight to the trash, turn left to the trash, turn right, turn left to the bathroom.
- go past the elevator, turn right, go through the doors, turn right to the microwave.
- go towards the fountain, turn right, go through the doors, turn left.
- turn right, go past the whiteboard, go towards the desk.
- go straight towards the desk, turn right, turn right to the tank.
- turn right, turn left, go through a door, go straight past the water fountain, turn right to the elevator.
- turn right, go straight past the couches, go towards the cabinets, turn left to the table, turn right, walk straight to the door.
- go past the water fountain, go through a door, turn left, turn right to the whiteboard.
- turn left, go past the bathroom, go straight through a door, turn right towards the desk.
- turn left, go through the intersection, turn left to the mailbox.
- go straight towards the microwave, turn left to the door, go straight past the tank, turn left to the intersection.

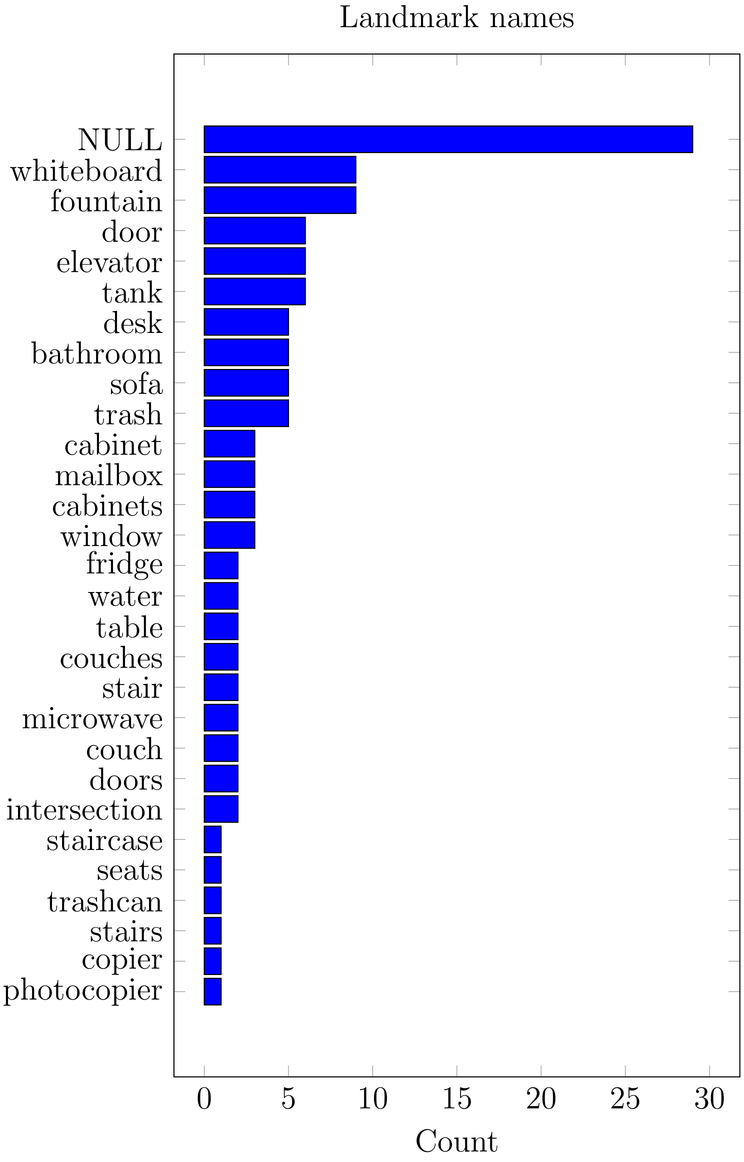
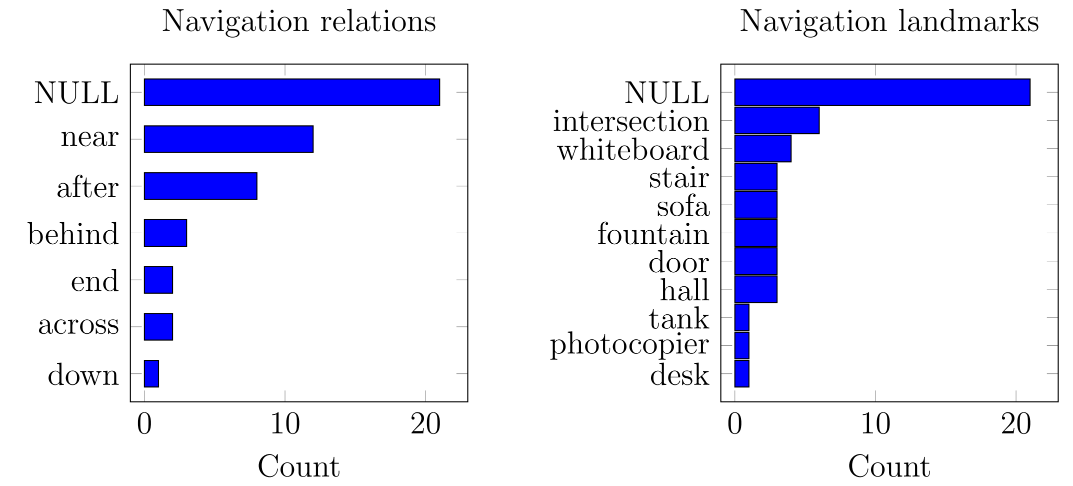


Figure A.3: Landmarks used in the basic corpus of directions. The basic corpus refers to many landmarks, some of which are not actually objects in the world (e.g., a "couch" is labeled as a "sofa" in the map). NULL means there is no landmark mentioned in the direction.

A.2 Corpus of Complex Natural Language Directions

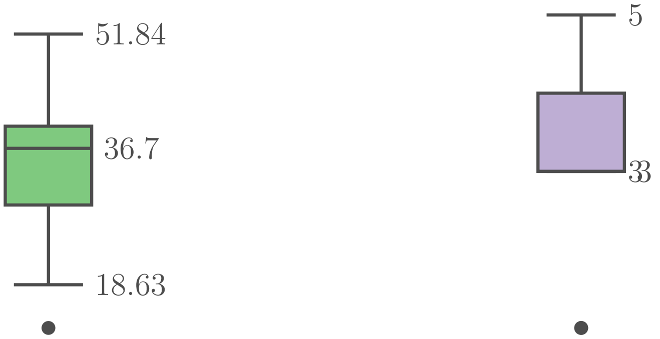
In addition to the basic corpus presented above, we created a corpus of more complex direction and used the combined corpora for the results presented in Section 6.3. These 15 directions contain additional navigation information that is useful for following the directions, generally references to other landmarks in the world. We show in Figure A.4 the navigation landmarks and relations. These are used by the policy as described in Chapter 5. Additionally, Figure A.5 shows statistics for the total distance each direction travels and the number of SDCs per direction across this corpus. The directions in the complex corpus are similar to the directions in the basic corpus: they travel a similar distance through the environment and contain a sequence of multiple SDCs.



(a) Navigation relation used in the complex direction (each references an object).

(b) Navigation landmark used in the complex direction.

Figure A.4: The complex direction contain additional navigation information the belief space policy can use to follow the direction. This consists of a navigation landmark and relation the policy can use to locate the goal landmark (e.g., "go to the door after the water fountain"). The navigation landmarks are all objects contained in the map.



(a) Path length statistics (m).

(b) Number of SDCs per direction.

Figure A.5: Distribution of path lengths and number of SDCs for directions in the complex corpus. (15 directions)

List of complex directions

Below is the complete list of 15 directions used in the complex corpus:

- go to the door across from the whiteboard.
- turn towards the trash near the stairs, turn left to the table, turn right, turn left to the door across from the sofa.
- turn towards the door near the intersection, turn right, go to the door near the desk.
- go through the door after the intersection.
- go past the window, go near the door that is after the whiteboard, turn towards the mailbox.
- go to the doors near the whiteboard, go past the fountain, turn right to the elevator.
- go to the doors near the tank, turn right, go straight to the refrigerator down the hall.
- turn left to the trash, go through the intersection, go towards the mailbox behind the stairs, turn right to the door near the intersection.
- go to the door after the water fountain, turn right, go straight to the cabinet.
- turn right, turn left to the phone near the door, go past the water fountain, go

towards the sofa behind the door.

- go towards the intersection, turn left, go through the door after the intersection, turn left, go to the window near the sofa.
- turn towards the door after the intersection, turn left to the door near the fountain, go to the elevator near the sofa.
- go towards the microwave after the photocopier, turn left, go to the door that is after the whiteboard, turn left to the tank near the intersection.
- go towards the desk at the end of the hall, turn right towards the intersection, turn towards the whiteboard near the door, go straight to the door that is after the water fountain.
- go towards the table behind the stairs, turn right, turn left to the sofa, turn right to the door at the end of the hall.

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, 2004. 2.2

[2] Pieter Abbeel, Adam Coates, and Andrew Y. Ng. Autonomous Helicopter Aerobatics through Apprenticeship Learning. *The International Journal of Robotics Research*, 2010. 2.2

[3] Philip E Agre and David Chapman. What are plans for? *Robotics and Autonomous Systems*, 1990. 2.1.3

[4] Abhishek Anand, Hema Swetha Koppula, Thorsten Joachims, and Ashutosh Saxena. Contextually Guided Semantic Labeling and Search for 3D Point Clouds. *International Journal of Robotics Research*, 2012. 2.5

[5] A Anderson, M Bader, E Bard, E Boyle, G M Doherty, S Garrod, S Isard, J Kowtko, J McAllister, J Miller, C Sotillo, H S Thompson, and R Weinert. The HCRC Map Task Corpus. *Language and Speech*, 1991. 2.1.1, 2.1.1

[6] Brian D. O. Anderson and John B. Moore. *Optimal control: linear quadratic methods*. 1990. 2.2

[7] John R Anderson, Daniel Bothell, Michael D Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin. An integrated theory of the mind. *Psychological review*, 2004. 8.1

[8] Jacob Andreas and Dan Klein. Grounding Language with Points and Paths in Continuous Spaces. In *Conference on Natural Language Learning*, 2014. 2.1.1, 2.1.4

[9] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 2009. 2.2

[10] Brenna D. Argall, Brett Browning, and Manuela M. Veloso. Teacher feedback to scaffold and refine demonstrated motion primitives on a mobile robot. *Robotics and Autonomous Systems*, 2011. 2.2

[11] Chris L Baker, Joshua B Tenenbaum, and Rebecca R Saxe. Bayesian models of human action understanding. In *Advances in Neural Information Processing Systems*, 2006. 2.3

[12] Chris L Baker, Rebecca Saxe, and Joshua B Tenenbaum. Action understanding as inverse planning. *Cognition*, 2009. 2.3

[13] Chris L Baker, Rebecca R Saxe, and Joshua B Tenenbaum. Bayesian Theory of Mind: Modeling Joint Belief-Desire Attribution. In *Conference of the Cognitive Science Society*, 2009. 2.3

[14] Emanuele Bastianelli, Domenico Daniele Bloisi, Roberto Capobianco, Fabrizio Cossu, Guglielmo Gemignani, Luca Iocchi, and Daniele Nardi. On-line Semantic Mapping. In *International Conference on Advanced Robotics*, 2013. 2.5

[15] Andrea Bauer, Klaas Klasing, Georgios Lidoris, Quirin Mühlbauer, Florian Rohrmüller, Stefan Sosnowski, Tingting Xu, Kolja Kühnlenz, Dirk Wollherr, and Martin Buss. The Autonomous City Explorer: Towards Natural Human-Robot Interaction in Urban Environments. *International Journal of Social Robotics*, 2009. 2.1.3

[16] Joydeep Biswas. *Vector Map-Based, Non-Markov Localization for Long-Term Deployment of Autonomous Mobile Robots*. PhD thesis, Carnegie Mellon University, 2014. 7.1

[17] Joydeep Biswas and Manuela Veloso. WiFi localization and navigation for autonomous indoor mobile robots. In *International Conference on Robotics and Automation*, 2010. 3.2

[18] Joydeep Biswas and Manuela M Veloso. Localization and Navigation of the CoBots Over Long-term Deployments. *The International Journal of Robotics Research*, 2013. 3.2

[19] Dan Bohus and Eric Horvitz. On the Challenges and Opportunities of Physically Situated Dialog. In *Dialog with Robots: AAAI Fall Symposium*, 2010. 2.1.3

[20] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. 1994. 2.2

[21] S R K Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *International Joint Conference on Natural Language Processing*, 2009. 2.1.3

[22] S R K Branavan, Luke Zettlemoyer, and Regina Barzilay. Reading Between the Lines: Learning to Map High-level Instructions to Commands. In *Association for Computational Linguistics*, 2010. 2.1.3

[23] Emma Brunskill, Thomas Kollar, and Nicholas Roy. Topological Mapping Using Spectral Clustering and Classification. In *International Conference on*

Intelligent Robots and Systems, 2007. 2.5

[24] Guido Bugmann, Ewan Klein, Stanislao Lauria, and Theocharis Kyriacou. Corpus-Based Robotics: A Route Instruction Example. In *Intelligent Autonomous Systems*, 2004. 2.1.2, 2.1.4

[25] Rehj Cantrell, Matthias Scheutz, Paul Schermerhorn, and Xuan Wu. Robust spoken instruction understanding for HRI. In *Human-Robot Interaction*, 2010. 2.1.3

[26] Rehj Cantrell, Kartik Talamadupula, Paul Schermerhorn, J Benton, Subbarao Kambhampati, and Matthias Scheutz. Tell me when and why to do it! Run-time Planner Model Updates Via Natural Language Instruction. In *Human-Robot Interaction*, 2012. 2.1.3

[27] David Chapman. *Vision, Instruction and Action*. PhD thesis, Massachusetts Institute of Technology, 1990. 2.1.3

[28] David L Chen and Raymond J Mooney. Training a Multilingual Sportscaster: Using Perceptual Context to Learn Language. *Journal of Artificial Intelligence Research*, 2010. 9.3

[29] David L Chen and Raymond J Mooney. Learning to Interpret Natural Language Navigation Instructions from Observations. In *AAAI Conference on Artificial Intelligence*, 2011. 2.1.2, 2.1.4

[30] Sonia Chernova and Manuela Veloso. Confidence-Based Multi-Robot Learning from Demonstration. *International Journal of Social Robotics*, 2010. 2.2

[31] Adam Coates, Pieter Abbeel, and Andrew Y. Ng. Learning for control from multiple demonstrations. In *International Conference on Machine Learning*, 2008. 2.2

[32] Adam Coates, Pieter Abbeel, and Andrew Y. Ng. Apprenticeship learning for helicopter control. *Communications of the ACM*, 2009. 2.2

[33] Brian Coltin, Joydeep Biswas, Dean Pomerleau, and Manuela Veloso. Effective Semi-autonomous Telepresence. In *Proceedings of the RoboCup Symposium*, 2011. 3.2

[34] Koby Crammer and Yoram Singer. On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines. *Journal of Machine Learning Research*, 2002. 4.1, 6.2

[35] Hal Daumé, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine Learning*, 2009. 2.2

[36] Guillaume de Chambrier and Aude Billard. Learning search polices from humans in a partially observable context. *Robotics and Biomimetics*, 2014. 2.3

[37] Robert Dean. Common world model for unmanned systems. In *Proc. of SPIE*,

2013. 8.1

[38] Robin Deits, Stefanie Tellex, Pratiksha Thaker, Dimitar Simeonov, Thomas Kollar, and Nicholas Roy. Clarifying Commands with Information-Theoretic Human-Robot Dialog. *Journal of Human-Robot Interaction*, 2013. 2.1.3

[39] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. *Statistics and Computing*, 2000. 5.3, 5.3

[40] Felix Duvallet and Anthony Stentz. Imitation Learning for Task Allocation. In *International Conference on Intelligent Robots and Systems*, 2010. 2.2

[41] Felix Duvallet, Thomas Kollar, and Anthony Stentz. Imitation Learning for Natural Language Direction Following through Unknown Environments. In *International Conference on Robotics and Automation*, 2013. 3.1

[42] Felix Duvallet, Matthew R Walter, Thomas Howard, Jean Oh, Seth Teller, Nicholas Roy, and Anthony Stentz. Inferring Maps and Behaviors from Natural Language Instructions. In *International Symposium on Experimental Robotics*, 2014. 5, 7

[43] Juraj Dzifcak, Matthias Scheutz, Chitta Baral, and Paul Schermerhorn. What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *International Conference on Robotics and Automation*, 2009. 2.1.3

[44] Juan Fasola and Maja J Mataric. Modeling dynamic spatial relations with global properties for natural language-based human-robot interaction. In *International Symposium on Robot and Human Interactive Communication*, 2013. 2.1.1

[45] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. 1998. 2.1.1, 3.4

[46] Dave Ferguson and Anthony Stentz. Field D*: An interpolation-based path planner and replanner. In *International Symposium on Robotics Research*, 2005. 3.2

[47] David Ferguson and Anthony Stentz. Using interpolation to improve path planning: The Field D* algorithm. *Journal of Field Robotics*, 2006. 3.2

[48] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981. 8.1

[49] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Active Markov localization for mobile robots, 1998. 2.4

[50] Stephen Friedman, Hanna Pasula, and Dieter Fox. Voronoi Random Fields: Extracting the Topological Structure of Indoor Environments via Place Labeling.

In *International Joint Conference on Artificial Intelligence*, 2007. 2.5

[51] Guglielmo Gemignani, Daniele Nardi, Domenico Daniele Bloisi, Roberto Capobianco, and Luca Iocchi. Interactive Semantic Mapping: Experimental Evaluation. In *International Symposium on Experimental Robotics*, 2014. 2.5

[52] Robert Goeddel and Edwin Olson. DART: A particle-based method for generating easy-to-follow directions. In *International Conference on Intelligent Robots and Systems*, 2012. 2.1.3

[53] Dave Golland, Percy Liang, and Dan Klein. A Game-Theoretic Approach to Generating Spatial Descriptions. *Computational Linguistics*, 2010. 2.1.3

[54] Stevan Harnad. The Symbol Grounding Problem. *Physica D: Nonlinear Phenomena*, 1990. 2.1

[55] Sachithra Hemachandra, Thomas Kollar, Nicholas Roy, and Seth Teller. Following and interpreting narrated guided tours. *International Conference on Robotics and Automation*, 2011. 7

[56] Sachithra Hemachandra, Matthew R Walter, Stefanie Tellex, and Seth Teller. Learning Spatial-Semantic Representations from Natural Language Descriptions and Scene Classifications. In *International Conference on Robotics and Automation*, 2014. 2.5, 7.3

[57] Sachithra Hemachandra, Felix Duvallet, Thomas M. Howard, Nicholas Roy, Anthony Stentz, and Matthew R Walter. Learning Models for Following Natural Language Directions in Unknown Environments. In *International Conference on Robotics and Automation*, 2015. 5, 7

[58] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *IEEE Signal Processing Magazine*, 2012. 9.3

[59] Thomas M Howard, Istvan Chung, Oron Propp, Matthew R Walter, and Nicholas Roy. Efficient Natural Language Interfaces for Assistive Robots. In *IROS Workshop on Rehabilitation and Assistive Robotics*, 2014. 5.2, 8.1

[60] Thomas M. Howard, Stefanie Tellex, and Nicholas Roy. A natural language planner interface for mobile manipulators. In *International Conference on Robotics and Automation*, 2014. 2.1.1, 2.1.4, 5.2

[61] Albert S Huang, Stefanie Tellex, Abraham Bachrach, Thomas Kollar, Deb Roy, and Nicholas Roy. Natural Language Command of an Autonomous Micro-Air Vehicle. In *International Conference on Intelligent Robots and Systems*, 2010. 2.1.1, 3.1

[62] Ray Jackendoff. *Semantics and Cognition*. 1983. 3.1

[63] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998. 2.3

[64] Michael Kaess. AprilTags C++ library. `http://people.csail.mit.edu/kaess/apriltags/`. 7.2, 7.3

[65] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 2008. 5.3

[66] Rudolf Emil Kalman. When is a linear control system optimal? *Journal of Fluids Engineering*, 1964. 2.2

[67] Sertac Karaman and Emilio Frazzoli. Sampling-based Algorithms for Optimal Motion Planning. *The International Journal of Robotics Research*, 2011. 3.2

[68] Andrej Karpathy and Li Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions. Technical report, Stanford University, 2014. 9.3

[69] Scott Kiesel and Wheeler Ruml. Planning Under Temporal Uncertainty Using Hindsight Optimization. In *ICAPS Planning and Robotics Workshop*, 2014. 2.3

[70] Scott Kiesel, Ethan Burns, Wheeler Ruml, J Benton, and Frank Kreimendahl. Open World Planning for Robots via Hindsight Optimization. In *ICAPS Planning and Robotics Workshop*, 2013. 2.3

[71] T Kollar and N Roy. Using reinforcement learning to improve exploration trajectories for error minimization. In *International Conference on Robotics and Automation*, 2006. 2.4

[72] Thomas Kollar. *Learning to Understand Spatial Language for Robotic Navigation and Mobile Manipulation*. PhD thesis, Massachusetts Institute of Technology, 2011. 2.1.1, 3.1, 3.4, 3.4, 4.5.1

[73] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward Understanding Natural Language Directions. In *International Conference on Human-Robot Interaction*, 2010. 2.1.1, 2.1.4, 3.1

[74] Thomas Kollar, Stefanie Tellex, and Nicholas Roy. A Discriminative Model for Understanding Natural Language Route Directions. In *AAAI Fall Symposium Series*, 2010. 2.1.1, 3.1

[75] Thomas Kollar, Vittorio Perera, Daniele Nardi, and Manuela Veloso. Learning environmental knowledge from task-based human-robot dialog. *International Conference on Robotics and Automation*, 2013. 2.5, 9.3

[76] Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. From structured english to robot motion. In *International Conference on Intelligent Robots and Systems*, 2007. 2.1.1

[77] Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. Translating

Structured English to Robot Controllers. *Advanced Robotics*, 2008. 2.1.1, 2.1.4

[78] Jayant Krishnamurthy and Thomas Kollar. Jointly Learning to Parse and Perceive: Connecting Natural Language to the Physical World. *Transactions of the Association for Computational Linguistics*, 2013. 9.3

[79] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, 2012. 9.3

[80] Benjamin Kuipers. Spatial semantic hierarchy. *Artificial Intelligence*, 2000. 2.5

[81] Theocharis Kyriacou, Guido Bugmann, and Stanislao Lauria. Vision-based urban navigation procedures for verbally instructed robots. *Robotics and Autonomous Systems*, 2005. 2.1.2, 2.1.4

[82] Kevin Lai, Liefeng Bo, and Dieter Fox. Unsupervised Feature Learning for 3D Scene Labeling. In *International Conference on Robotics and Automation*, 2014. 2.5

[83] Barbara Landau and Ray Jackendoff. What and where in spatial language and spatial cognition. *Behavioral and Brain Sciences*, 1993. 3.1

[84] Christian Landsiedel, Roderick De Nijs, Kolja Kuhnlenz, Dirk Wollherr, and Martin Buss. Route description interpretation on automatically labeled robot maps. In *International Conference on Robotics and Automation*, 2013. 2.1.1

[85] Stanislao Lauria, Guido Bugmann, Theocharis Kyriacou, and Ewan Klein. Mobile robot programming using natural language. *Robotics and Autonomous Systems*, 2002. 2.1.2

[86] Michael Levit and Deb Roy. Interpretation of spatial language in a map navigation task. *Systems, Man, and Cybernetics*, 2007. 2.1.1, 2.1.4

[87] Constantine Lignos, Vasumathi Raman, Cameron Finucane, Mitchell Marcus, and Hadas Kress-Gazit. Provably correct reactive control from natural language. *Autonomous Robots*, 2014. 2.1.1

[88] Michael L Littman, Anthony R Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In *International Conference on Machine Learning*, 1995. 2.3

[89] James MacGlashan, Monica Babes-Vroman, Marie DesJardins, Michael Littman, Smaranda Muresan, and Shawn Squire. Translating English to Reward Functions. Technical report, Computer Science Department Brown University, 2014. 2.1.1

[90] James Macglashan, Michael Littman, Robert Loftin, Bei Peng, David Roberts, and Matthew E Taylor. Training an Agent to Ground Commands with Reward and Punishment. In *AAAI Machine Learning for Interactive Systems Workshop*,

2014. 2.1.1

[91] Matthew MacMahon. MARCO: A Modular Architecture for Following Route Instructions. In *AAAI Workshop on Modular Construction of Human-like Intelligence*, 2005. 2.1.2

[92] Matthew MacMahon. *Following Natural Language Route Instructions*. PhD thesis, University of Texas at Austin, 2007. 2.1.2, 2.1.2

[93] Matthew MacMahon, Brian Stankiewicz, and Benjamin Kuipers. Walk the Talk: Connecting Language, Knowledge, and Action in Route Instructions. In *National Conference on Artificial Intelligence*, 2006. 2.1.2, 2.1.4

[94] A Makarenko, S B Williams, F Bourgault, and H F Durrant-Whyte. An experiment in integrated exploration. In *International Conference on Intelligent Robots and Systems*, 2002. 2.4, 3.2

[95] Christian Mandel, Udo Frese, and Thomas Rofer. Robot navigation based on the mapping of coarse qualitative route descriptions to route graphs. In *International Conference on Intelligent Robots and Systems*, 2006. 2.1.1

[96] Matthew Marge and AI Rudnicky. Comparing spoken language route instructions for robots across environment representations. In *Proceedings of SIGdial*, 2010. 2.1.3

[97] Matthew Marge and Alexander I Rudnicky. The TeamTalk Corpus: Route Instructions in Open Spaces. In *Workshop on Grounding Human-Robot Dialog for Spatial Tasks*, 2011. 2.1.3

[98] Cynthia Matuszek, Dieter Fox, and Karl Koscher. Following directions using statistical machine translation. In *Human-Robot Interaction*, 2010. 2.1.2

[99] Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. Learning to Parse Natural Language Commands to a Robot Control System. In *International Symposium on Experimental Robotics*, 2012. 2.1.2, 2.1.4

[100] Çetin Meriçli, Steven D Klee, Jack Paparian, and Manuela Veloso. An Interactive Approach for Situated Task Specification through Verbal Instructions. In *International Conference on Autonomous Agents and Multiagent Systems*, 2014. 2.1.3

[101] George Miller. WordNet: An on-line lexical database. *International Journal of Lexicography*, 1990. 2.1.1, 3.4

[102] Dipendra K Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. Tell Me Dave: Context-Sensitive Grounding of Natural Language to Manipulation Instructions. In *Robotics: Science and Systems*, 2014. 2.1.3

[103] Raymond J Mooney. Learning to Connect Language and Perception. In *AAAI Conference on Artificial Intelligence*, 2008. 9.3

[104] Oscar Martinez Mozos, Rudolph Triebel, Patric Jensfelt, Axel Rottmann, and Wolfram Burgard. Supervised semantic labeling of places using information extracted from sensor data. *Robotics and Autonomous Systems*, 2007. 2.5

[105] Krikamol Muandet, Bernhard Schölkopf, Kenji Fukumizu, and Francesco Dinuzzo. Learning from Distributions via Support Measure Machines. In *Advances in Neural Information Processing Systems*, 2012. 6.1

[106] R Müller, T Röfer, Axel Lankenau, A Musto, K Stein, and A. Coarse qualitative descriptions in robot navigation. *Spatial Cognition II*, 2000. 2.1.1

[107] Daniel Munoz, J. Andrew Bagnell, and Martial Hebert. Stacked Hierarchical Labeling. In *European Conference on Computer Vision*, 2010. 8.1

[108] Andrew Y. Ng and Stuart Russell. Algorithms for Inverse Reinforcement Learning. In *International Conference on Machine Learning*, 2000. 2.2

[109] Andreas Nüchter and Joachim Hertzberg. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 2008. 2.5

[110] Robert B Ochsman and Alphonse Chapanis. The effects of 10 communication modes on the behavior of teams during co-operative problem-solving. *International Journal of Man-Machine Studies*, 1974. 2.1

[111] Jean Oh, Arne Suppe, Felix Duvallet, Abdeslam Boularias, Jerry Vinokurov, Luis Navarro-Serment, Oscar Romero, Robert Dean, Christian Lebiere, Martial Hebert, and Anthony Stentz. Toward Mobile Robots Reasoning Like Humans. In *AAAI Conference on Artificial Intelligence*, 2015. 7, 8

[112] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *International Conference on Robotics and Automation*, 2011. 7.2, 7.3, 7.3

[113] Stefan Osswald, H Kretzschmar, Wolfram Burgard, and Cyrill Stachniss. Learning to Give Route Directions from Human Demonstrations. In *International Conference on Robotics and Automation*, 2014. 2.1.3

[114] R Platt, R Tedrake, L Kaelbling, and T Lozano-Perez. Belief space planning assuming maximum likelihood observations. In *Robotics: Science and Systems*, 2010. 2.3

[115] Robert Platt, Leslie Kaelbling, Tomas Lozano-Perez, and Russ Tedrake. Simultaneous Localization and Grasping as a Belief Space Control Problem. In *International Symposium on Robotics Research*, 2011. 2.3

[116] Robert Platt, Leslie Kaelbling, Tomas Lozano-Perez, and Russ Tedrake. Non-Gaussian Belief Space Planning: Correctness and Complexity. In *International Conference on Robotics and Automation*, 2012. 2.3

[117] Dean Pomerleau. ALVINN: An Autonomous Land Vehicle In a Neural Network. In *Advances in Neural Information Processing Systems*, 1989. 2.2

[118] S. Prentice and N. Roy. The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance. *The International Journal of Robotics Research*, 2009. 2.3

[119] Andrzej Pronobis, O. Martinez Mozos, B. Caputo, and P. Jensfelt. Multi-modal Semantic Place Classification. *The International Journal of Robotics Research*, 2010. 2.5

[120] Vasumathi Raman, Constantine Lignos, Cameron Finucane, Kenton C. T. Lee, Mitch Marcus, and Hadas Kress-Gazit. Sorry Dave, I'm Afraid I Can't Do That: Explaining Unachievable Robot Tasks Using Natural Language. In *Robotics: Science and Systems*, 2013. 2.1.1, 2.1.3

[121] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. Maximum Margin Planning. In *International Conference on Machine Learning*, 2006. 2.2, 4.1, 4.1, 6.2, 8.1

[122] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. (Online) Subgradient Methods for Structured Prediction. In *International Conference on Artificial Intelligence and Statistics*, 2007. 2.2

[123] Nathan D. Ratliff, David Silver, and J. Andrew Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 2009. 2.2

[124] Nathan D. Ratliff, Brian Ziebart, Kevin Peterson, J. Andrew Bagnell, Martial Hebert, Anind K Dey, and Siddhartha S. Srinivasa. Inverse Optimal Heuristic Control for Imitation Learning. In *International Conference on Artificial Intelligence and Statistics*, 2009. 2.2

[125] Stephanie Rosenthal, Joydeep Biswas, and Manuela Veloso. An effective personal mobile robot agent through symbiotic human-robot interaction. In *Autonomous Agents and Multi-Agent Systems*, 2010. 7

[126] Stéphane Ross and J. Andrew Bagnell. Efficient Reductions for Imitation Learning. In *International Conference on Artificial Intelligence and Statistics*, 2010. 2.2, 4.3

[127] Stéphane Ross and J. Andrew Bagnell. Agnostic System Identification for Model-Based Reinforcement Learning. In *International Conference on Machine Learning*, 2012. 2.2

[128] Stephane Ross and J. Andrew Bagnell. Reinforcement and Imitation Learning via Interactive No-Regret Learning. Technical report, 2014. 2.2

[129] Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *International Conference on Artificial Intelligence and Statistics*, 2011. 2.2,

4.3, 6.2

[130] Stephane Ross, Daniel Munoz, Martial Hebert, and J. Andrew Bagnell. Learning message-passing inference machines for structured prediction. In *Conference on Computer Vision and Pattern Recognition*, 2011. 2.2

[131] Stephane Ross, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J. Andrew Bagnell, and Martial Hebert. Learning Monocular Reactive UAV Control in Cluttered Natural Environments. In *International Conference on Robotics and Automation*, 2013. 2.2

[132] Stephane Ross, Jiaji Zhou, Yison Yue, Debadeepta Dey, and J. Andrew Bagnell. Learning Policies for Contextual Submodular Prediction. In *International Conference on Machine Learning*, 2013. 2.2

[133] Nicholas Roy, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Coastal Navigation - Mobile Robot Navigation with Uncertainty in Dynamic Environments. In *International Conference on Robotics and Automation*, 1999. 2.3

[134] Alexander I. Rudnicky. Mode preference in a simple data-retrieval task. In *Proceedings of the Workshop on Human Language Technology*, 1993. 2.1

[135] Michael C. Runge, Sarah J. Converse, and James E. Lyons. Which uncertainty? Using expert elicitation and expected value of information to design an adaptive program. *Biological Conservation*, 2011. 2.4

[136] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *International Conference on Robotics and Automation*, 2011. 8.1

[137] Paul E Rybski, Kevin Yoon, Jeremy Stolarz, and Manuela M Veloso. Interactive Robot Task Training through Dialog and Demonstration. In *Human-Robot Interaction*, 2007. 2.1.3

[138] Nobuyuki Shimizu and Andrew Haas. Learning to follow navigational route instructions. In *International Joint Conference on Artificial Intelligence*, 2009. 2.1.1, 2.1.4

[139] David Silver, J Andrew Bagnell, and Anthony Stentz. High Performance Outdoor Navigation from Overhead Data using Imitation Learning. In *Robotics: Science and Systems*, 2008. 2.2

[140] David Silver, J Andrew Bagnell, and Anthony Stentz. Perceptual Interpretation for Autonomous Navigation through Dynamic Imitation Learning. In *International Symposium of Robotics Research*, 2009. 2.2

[141] David Silver, J. Andrew Bagnell, and Anthony Stentz. Learning from Demonstration for Autonomous Navigation in Complex Unstructured Terrain. *The International Journal of Robotics Research*, 2010. 2.2

[142] David Silver, J. Andrew Bagnell, and Anthony Stentz. Learning Autonomous

Driving Styles and Maneuvers from Expert Demonstration. In *International Symposium on Experimental Robotics*, 2012. 2.2

[143] R Sim and N Roy. Global A-Optimal Robot Exploration in SLAM. In *International Conference on Robotics and Automation*, 2005. 2.4

[144] Dimitar Simeonov, Stefanie Tellex, Thomas Kollar, and Nicholas Roy. Toward Interpreting Spatial Language Discourse with Grounding Graphs. In *Workshop on Grounding Human-Robot Dialog for Spatial Tasks*, 2011. 3.1

[145] Reid Simmons, Dani Goldberg, Adam Goode, Michael Montemerlo, Nicholas Roy, Brennan Sellner, Chris Urmson, Alan Schultz, Myriam Abramson, William Adams, Amin Atrash, Magda Bugajska, Michael Coblenz, Matthew MacMahon, Dennis Perzanowski, Ian Horswill, Robert Zubek, David Kortenkamp, Bryn Wolfe, Tod Milam, and Bruce Maxwell. GRACE: An Autonomous Robot for the AAAI Robot Challenge. *AAAI Magazine*, 2003. 2.1.3

[146] Marjorie Skubic, Dennis Perzanowski, Samuel Blisard, Alan Schultz, William Adams, Magda Bugajska, and Derek Brock. Spatial language for human-robot dialogs. In *Systems, Man, and Cybernetics*, 2004. 2.1.3

[147] Alex Smola, Arthur Gretton, Le Song, and B Schölkopf. A Hilbert space embedding for distributions. In *Algorithmic Learning Theory*, 2007. 6.1, 6.1

[148] L Song, B Boots, and S M Siddiqi. Hilbert space embeddings of hidden Markov models. In *International Conference on Machine Learning*, 2010. 6.1

[149] Le Song, Jonathan Huang, Alex Smola, and Kenji Fukumizu. Hilbert space embeddings of conditional distributions with applications to dynamical systems. In *International Conference on Machine Learning*, 2009. 6.1

[150] Cyrill Stachniss, Dirk Hähnel, and Wolfram Burgard. Exploration with active loop-closing for FastSLAM. In *International Conference on Intelligent Robots and Systems*, 2004. 2.4

[151] Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. Information Gain-based Exploration Using Rao-Blackwellized Particle Filters. In *Robotics: Science and Systems*, 2005. 2.4, 3.2

[152] Cyrill Stachniss, Oscar Martinez-Mozos, Axel Rottmann, and Wolfram Burgard. Semantic labeling of places. In *International Symposium of Robotics Research*, 2005. 2.5

[153] Anthony Stentz. Optimal and efficient path planning for partially-known environments. In *International Conference on Robotics and Automation*, 1994. 3.2

[154] Yichao Sun, Brian Coltin, and Manuela Veloso. Interruptible Autonomy: Towards Dialog-Based Robot Task Management. In *AAAI Workshop on Intelligent*

Robotic Systems, 2013. 2.1.3

[155] Leonard Talmy. The fundamental system of spatial schemas in language. *From Perception to Meaning: Image Schemas in Cognitive Linguistics*, 2005. 3.1

[156] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems*, 2003. 2.2

[157] Stefanie Tellex. *Natural Language and Spatial Reasoning*. PhD thesis, Massachusetts Institute of Technology, 2010. 2.1.1, 3.1, 3.4

[158] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation. In *National Conference on Articial Intelligence*, 2011. 2.1.1, 2.1.4, 3.1, 8.1

[159] Stefanie Tellex, Pratiksha Thaker, Robin Deits, Dimitar Simeonov, Thomas Kollar, and Nicholas Roy. Toward Information Theoretic Human-Robot Dialog. In *Robotics: Science and Systems*, 2012. 2.1.3

[160] Stefanie Tellex, Ross A Knepper, Adrian Li, Daniela Rus, and Nicholas Roy. Asking for Help Using Inverse Semantics. In *Robotics: Science and Systems*, 2014. 2.1.3

[161] M Tenorth, D Nyga, and M Beetz. Understanding and executing instructions for everyday manipulation tasks. In *International Conference on Robotics and Automation*, 2010. 2.1.3

[162] Sebastian Thrun. Toward a Framework for Human-Robot Interaction. *Human-Computer Interaction*, 2004. 2.1

[163] J. van den Berg, S. Patil, and R. Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *The International Journal of Robotics Research*, 2012. 2.3

[164] Manuela Veloso, Joydeep Biswas, Brian Coltin, Stephanie Rosenthal, Susana Brandao, Tekin Mericli, and Rodrigo Ventura. Symbiotic-Autonomous Service Robots for User-Requested Tasks in a Multi-Floor Building. In *IROS Workshop on Cognitive Assistive Systems: Closing the Action-Perception Loop*, 2012. 3.2, 7, 7.1

[165] Deepak Verma and Rajesh P. N. Rao. Goal-based imitation as probabilistic inference over graphical models. In *Advances in Neural Information Processing Systems*, 2005. 2.3

[166] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and Tell: A Neural Image Caption Generator. Technical report, Google, 2014. 9.3

[167] Adam Vogel and Dan Jurafsky. Learning to Follow Navigational Directions. In Association For Computational Linguistics, editor, *Association for Compu-*

tational Linguistics. Association for Computational Linguistics, 2010. 2.1.1, 2.1.4

[168] M. R. Walter, S. Hemachandra, B. Homberg, S. Tellex, and S. Teller. A framework for learning semantic maps from grounded natural language descriptions. *The International Journal of Robotics Research*, 2014. 2.5, 5.3, 9.3

[169] Tom Williams, Rehj Cantrell, Gordon Briggs, Paul Schermerhorn, and Matthias Scheutz. Grounding Natural Language References to Unvisited and Hypothetical Locations. In *AAAI Conference on Artificial Intelligence*, 2013. 2.5

[170] Terry Winograd. *Procedures as a Representation for Data in a Computer Program for Understanding Natural Language*. PhD thesis, Massachusetts Institute of Technology, 1971. 2.1.3

[171] Brian Yamauchi. Frontier-Based Exploration Using Multiple Robots. In *International Conference on Autonomous Agents*, 1998. 3.2

[172] Sungwook Yoon, Alan Fern, Robert Givan, and Subbarao Kambhampati. Probabilistic Planning via Determinization in Hindsight. In *AAAI Conference on Artificial Intelligence*, 2008. 2.3

[173] H Zender, O Mart, P Jensfelt, G M Kruijff, and W Burgard. Conceptual Spatial Representations for Indoor Mobile Robots. *Robotics and Autonomous Systems*, 2008. 2.5

[174] Brian Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K Dey. Maximum Entropy Inverse Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*, 2008. 2.2